What is signal analysis with Haar Matrices?

Idil Sahin

Dartmouth College

Abstract

This *what is?* article explores Haar matrices, wavelets, their application and importance for signal processing, image and audio compression.

1. Introduction

Haar matrices and the corresponding Haar wavelets are fundamental tools in signal processing and computer graphics. Before diving in, let's start with a motivating example:

Consider a black-and-white image. A 512 × 512 image assigns an intensity value to each pixel. (Although real images use integer intensities, we'll treat them as real numbers.) You may be familiar with downsampling, where a 512 × 512 image is reduced to 256 × 256 by grouping 2 × 2 pixel blocks and replacing each block with the average intensity of its pixels. This process can continue until we are left with a 1 × 1, whose intensity is the average of all original pixel values. This repeated subsampling is the basis of wavelets. (Klein, 2013)

Wavelets are especially valuable for **encoding long signals** where they enable compression into much shorter signals that retain enough information to be indistinguishable from the original (Vidakovic, 1999). Consider the four vectors $w_1, w_2, w_3, w_4 \in R^4$ given by:

$$w_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad w_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}, \quad w_3 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \quad w_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix}$$

Observe that these vectors are pairwise orthogonal, meaning their inner product is 0, as they are enough of them to span R^4, they form a basis. Let $\mathcal{W} = [w_1\ w_2\ w_3\ w_4]$ be the Haar basis, and let $U = [e_1\ e_2\ e_3\ e_4]$ be the canonical basis of $\mathbb{R}^4$. The change of basis matrix $W = P_{(WU)}$ from U to W is given by:

$$W = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix}$$

Observe that as $\mathcal{U}$ is the standard basis, this transformation is the same as $\mathcal{W}$ itself. Recall that the columns of W are mutually orthonormal up to a scalar factor. Now observe that:

$$W^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \qquad W^T W = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

Let $W^T W = D$ as it is a diagonal matrix. Subsequently the inverse of W is easily found by:

$$W^{-1} = D^{-1} W^T = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

With any signal $v = [v_1, v_2, v_3, v_4]$, we transform v into its coefficents $c = [c_1, c_2, c_3, c_4]$ over the Haar basis by calculating $c = W^{-1}v$.As an example, the the vector $v = [6, 4, 5, 1]$ over the basis $\mathcal{U}$ is transformed to $c = [c_1, c_2, c_3, c_4]$.

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 4 \\ 6 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ -1 \\ 2 \end{bmatrix}$$

Observe that $v = v_1 + v_2 + v_3 + v_4)/4$ is the average value of signal v, which corresponds to the background signal. Similarly, $c_3 = (v_1 + v_2 - v_3 - v_4)/4$ gives us coarse general details of v, $c_3 = (v_1 - v_2)/2$ gives us details about the first part of v and $c_4 = (v_3 - v_4)/2$ gives details about the second part of v.

In order to retrieve back our original signal v, we compute $v = Wc$, but we want to make this process as low-cost as possible. The trick to good compression involves *discarding some coefficients of* c (setting them to zero) to obtain a **compressed signal** c, while still preserving enough critical information so that the reconstructed signal $\hat{v} = W\hat{c}$ closely resembles the original. Thus, our process is as follows:

$$\text{input vector } v \longrightarrow \text{coefficents } c = W^{-1}v \longrightarrow \text{compressed } \hat{c} \longrightarrow \text{compressed } \hat{v} = W\hat{c}$$

This multiresolution property allows for signal compression by retaining only the most

significant coefficients. For instance, consider the signal:

$$u = (24, 22, 21, 5, 20, 5, 6, 8, 28, 11, 13),$$

whose Haar transform is:

$$\mathbf{c} = (20, 2, 0, 13, 0, 10, 0, 5, 2, 0, 1).$$

Since some coefficients in $\mathbf{c}$ are small (e.g., ≤ 2), we can compress $\mathbf{c}$ by setting them to zero:

$$\mathbf{c}_2 = (20, 0, 0, 13, 0, 10, 0, 5, 0, 0, 0).$$

The reconstructed signal becomes:

$$u_2 = (2, 2, 2, 2, 7, 3, 1, 1).$$

2. Haar Transform for Digital Images

The 2D Haar transform can be applied to a $2^m \times 2^n$ matrix A representing an image. The process involves transforming the rows and columns of A using the Haar transform matrices W_n and W_m. We first convert the *rows* of A to their haar coefficients using the matrix W_n^{-1} and then transform the columns of B using W_m^{-1}[1]. In our first step, to operate on the rows instead of the columns, we take the transpose $(W_n^{-1})^T$ and multiply by the right:

$$B = A(W_n^{-1})^T.$$

We then apply W_m^{-1} to the columns of B, which gives us our final matrix C:

$$C = W_m^{-1}B = W_m^{-1}A(W_n^{-1})^T.$$

In order to reconstruct our image back, we apply W_m to the columns of C and W_n^T to the rows of B, which gives us the original matrix A:

$$B = W_m C, \quad A = BW_n^T = W_m C W_n^T.$$

An example of a 2-D Haar transform is implemented in the PyWavelets library (Lee et al., 2019). This implementation decomposes an image into four components: approximation (cA), horizontal detail (cH), vertical detail (cV), and diagonal detail (cD). They are as follows:

- cA: Approximation (low-frequency components). This component contains the

[1]In practice, we do not need to explicitly compute the inverse matrices W_m^{-1} and W_n^{-1}. Instead, we use efficient algorithms based on averaging and differencing to perform the transformations–explained in more detail in Appendix B.

coarse, low-frequency information of the image (overall structure).

- cH: Horizontal detail (high-frequency components in the horizontal direction).
- cV: Vertical detail (high-frequency components in the vertical direction).
- cD: Diagonal detail (high-frequency components in the diagonal direction).

They correspond to the following two quadrants of the final transformation matrix C (Schlueter & Deuschle, 2014; Talukder & Harada, 2010):

$$C = \begin{bmatrix} cA & cH \\ cV & cD \end{bmatrix}.$$

Every quadrant except the target one is set to zero during the compression, eg.

$$C' = \begin{bmatrix} cA & 0 \\ 0 & 0 \end{bmatrix}.$$

for the approximation matrix. The end results of this process can be observed in the following two figures:

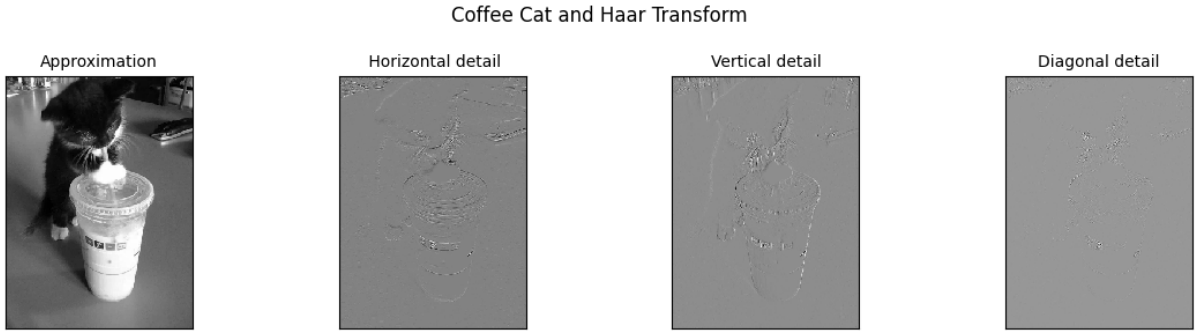

Figure 1: Haar transform of a low-contrast image.

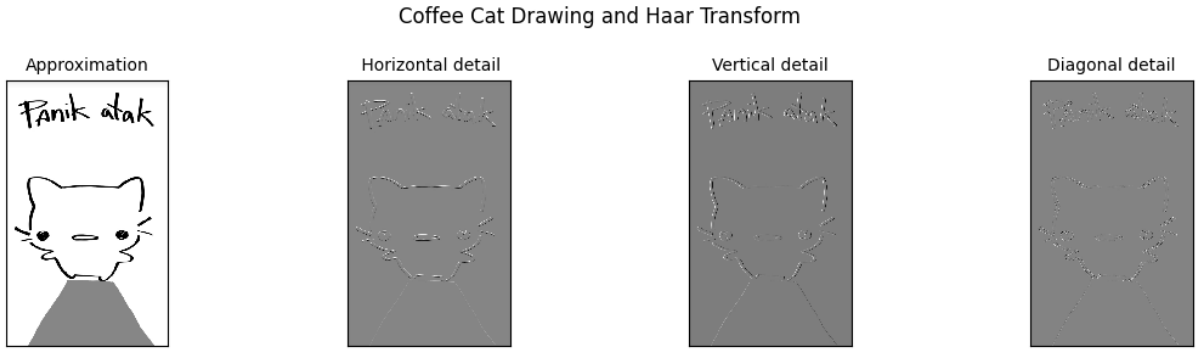

Figure 2: Haar transform of an higher-contrast image.

References

Klein, P. N. (2013). *Coding the matrix: Linear algebra through applications to computer science* [Google-Books-ID: 8PmPngEACAAJ]. Newtonian Press.

Lee, G., Gommers, R., Waselewski, F., Wohlfahrt, K., & O'Leary, A. (2019). PyWavelets: A python package for wavelet analysis. *Journal of Open Source Software*, 4(36), 1237. https://doi.org/10.21105/joss.01237

Schlueter, S., & Deuschle, C. (2014). Wavelet-based forecasting of ARIMA time series - an empirical comparison of different methods [Number: 1]. *Managerial Economics*, 15(1), 107–107. https://doi.org/10.7494/manage.2014.15.1.107

Talukder, K. H., & Harada, K. (2010, October 20). Haar wavelet based approach for image compression and quality assessment of compressed image. https://doi.org/10.48550/arXiv.1010.4084

Vidakovic, B. (1999). Discrete wavelet transformation [Section: 4]. In *Statistical modeling by wavelets* (pp. 101–117). John Wiley & Sons, Ltd.

Appendix

A. Code for the Haar Matrix Tranform

```
import pywt
import matplotlib.pyplot as plt
import numpy as np
import pywt.data
from PIL import Image


# transformation on a matrix
matrix = np.array([[63,2], [9,55]])
coeff_m = pywt.wavedec2(matrix, 'haar')
print(coeff_m)

# load image
original = "ice_cat.webp"

# read the image file using Pillow library
img = Image.open(original).convert('L') # Convert to grayscale
original = np.array(img) # Convert the image to a NumPy array

# wavelet transform of image, and plot approximation and details
titles = ['Approximation', ' Horizontal detail',
```

```
          'Vertical detail', 'Diagonal detail']

coeffs2 = pywt.dwt2(original, 'haar')

LL, (LH, HL, HH) = coeffs2
fig = plt.figure(figsize=(12, 3))

for i, a in enumerate([LL, LH, HL, HH]):
    ax = fig.add_subplot(1, 4, i + 1)
    ax.imshow(a, interpolation="nearest", cmap=plt.cm.gray)
    ax.set_title(titles[i], fontsize=10)
    ax.set_xticks([])
    ax.set_yticks([])


fig.suptitle("Coffee Cat and Haar Transform")
fig.tight_layout()
plt.show()


# reconstruct the image
reconstructed = pywt.idwt2(coeffs2, 'haar')

fig = plt.figure(figsize=(10, 5))


# plot reconstructed image
ax1 = fig.add_subplot(1, 2, 1)
ax1.imshow(reconstructed, interpolation="nearest", cmap=plt.cm.gray)
ax1.set_title("Reconstructed")
ax1.set_xticks([])
ax1.set_yticks([])

# plot original image
ax2 = fig.add_subplot(1, 2, 2)
ax2.imshow(original, interpolation="nearest", cmap=plt.cm.gray)
ax2.set_title("Original")
ax2.set_xticks([])
ax2.set_yticks([])

plt.tight_layout()
plt.show()
```

```
# check that reconstructed image is close to the original
np.testing.assert_allclose(original, reconstructed, atol=1e-13, rtol=1e-13)
```

B. Algorithms

This section I am leaving for my final draft! In case it would be nice and useful I would love to describe more how the inverses are computed.