Image Compression via Wavelets

GitHub repo

2020-21 Term 2 ESTR1005 Project Group 16

Chinese University of Hong Kong

GitHub repo: https://git.io/JOzxK

Theory of Haar Wavelets

Haar wavelets are the simplest type of wavelets, so we will focus on Haar wavelet transform to illustrate the idea of wavelets.
We will focus on the space $L^2(\mathbb{R})$, which is basically a vector space of functions with finite norm.

Motivation
Consider greyscale intensities:

110	100	120	140	...

We want to represent the above sequence by one function in order to analyze it mathematically.
Boxcar function

$$\phi(t) \coloneqq \begin{cases} 1 \text{ if } 0 \le t < 1 \\ 0 \text{ otherwise} \end{cases}$$

Using $\phi(t)$, we can express the above sequence by
$f(t) = 110\phi(t) + 100\phi(t-1) + 120\phi(t-2) + 140\phi(t-3) + \cdots$
In general, we have $f(t) = \sum_{k\in\mathbb{Z}} a_k \phi(t-k)$.

Haar Space V_j
We define the general Haar space V_j for integers j by

$$V_j \coloneqq \mathrm{Span}\{\phi(2^j t - k)\}_{k\in\mathbb{Z}} \cap L^2(\mathbb{R})$$

Here, $\{\phi(2^j t - k)\}_{k\in\mathbb{Z}}$ means $\{\ldots, \phi(2^j t+1), \phi(2^j t), \phi(2^j t-1), \ldots\}$.
V_j contains all possible functions of a 1D sequence of greyscale intensities!

Here we use $\phi(2^j t - k)$ instead of just $\phi(t-k)$ because to perform wavelet transforms, we need to handle functions with different resolutions. That is, we want functions that have constant intervals starting from every half-integer t, quarter-integer t, and so on.

For instance, consider $\phi(2t) = \begin{cases} 1 \text{ if } 0 \le t < \frac{1}{2}. \\ 0 \text{ otherwise} \end{cases}$

Using $\phi(2t)$ instead of $\phi(t)$ increases the resolution of $f(t)$ by doubling the pixels, i.e., constant intervals start from every half-integer t.

Approximating functions in $L^2(\mathbb{R})$ by functions in V_j
Functions in V_j can be used to approximate any function in $L^2(\mathbb{R})$, providing a theoretical basis for the further discussion on the Haar wavelet transform.
To find the best approximation, we need an orthonormal basis for V_j.
In fact, $\{2^{j/2}\phi(2^j t - k)\}_{k\in\mathbb{Z}}$ is an orthonormal basis for V_j.
Therefore, for $g(t) \in L^2(\mathbb{R})$, we have

$$proj_{V_j} g(t) = 2^j \sum_{k\in\mathbb{Z}} \langle \phi(2^j t - k), g(t) \rangle\, \phi(2^j t - k)$$

This can be seen as estimating a function by discrete pixels!
Intuitively, we can see that when $j \to \infty$, $proj_{V_j} g(t) \to g(t)$, so the accuracy can be adjusted by controlling j.

Approximating functions in V_1 by functions in V_0
Suppose a function $f_1(t) \in V_1$ is given by

$$f_1(t) = \sum_{k\in\mathbb{Z}} a_k \left(2^{1/2}\phi(2t-k)\right)$$

As $f_1(t) \in L^2(\mathbb{R})$,the projection $f_0(t) = proj_{V_0} f_1(t)$ is

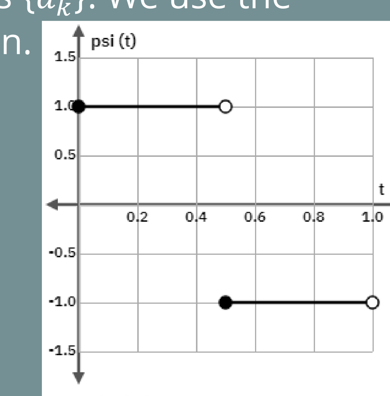
$$f_0(t) = \sum_{k\in\mathbb{Z}} \langle \phi(t-k), f_1(t) \rangle\, \phi(t-k) = \sum_{k\in\mathbb{Z}} b_k \phi(t-k)$$

where $b_k = \frac{\sqrt{2}}{2}(a_{2k} + a_{2k+1})$.

We call $g_0(t) = f_1(t) - f_0(t)$ the ***residual function***, i.e., the error of approximation.

We can find a formula relating g_0 and the original pixels $\{a_k\}$. We use the following *wavelet function* $\psi(t)$ to simplify the expression.

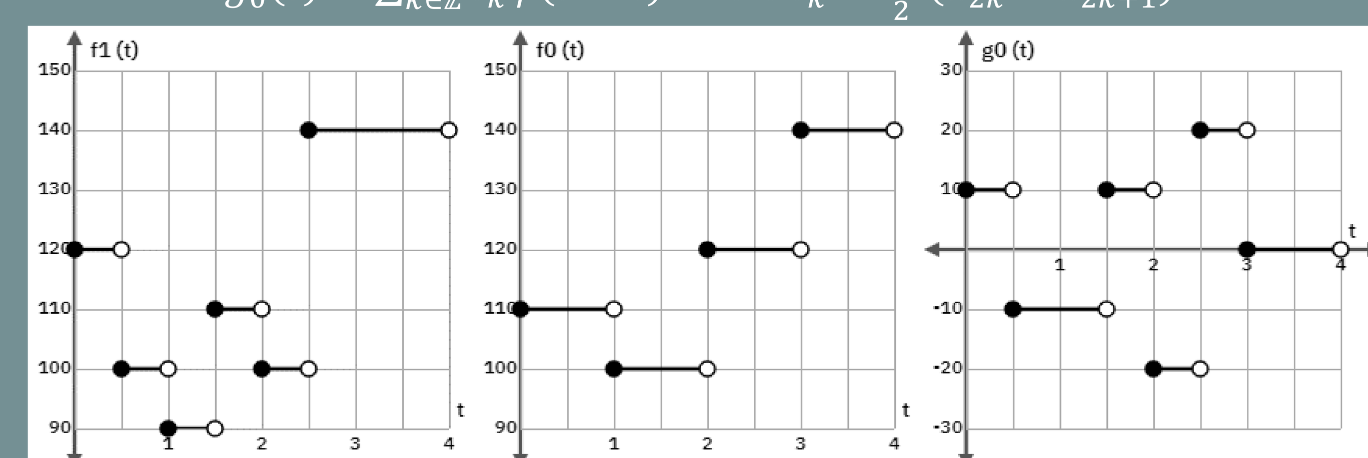
$$\psi(t) \coloneqq \phi(2t) - \phi(2t-1) = \begin{cases} 1 \text{ if } 0 \le t < \frac{1}{2} \\ -1 \text{ if } \frac{1}{2} \le t < 1 \\ 0 \text{ otherwise} \end{cases}$$



psi (t)

From the following example graphs, we can see

$$g_0(t) = \sum_{k\in\mathbb{Z}} c_k \psi(t-k) \text{ where } c_k = \frac{\sqrt{2}}{2}(a_{2k} - a_{2k+1})$$



f1 (t) f0 (t) g0 (t)

(In each integer interval, f_0 is the average of f_1, making g_0 a multiple of ψ.)
We can see why the function $\psi(t)$ is called a wavelet here:
Small waves like $\psi(t-k)$ can be used to model the error because of the nature of the approximation. Other wavelets work in a similar principle.

Haar Wavelet Space W_j
To simplify the notations, we will define:

$$\phi_{j,k}(t) \coloneqq 2^{j/2}\phi(2^j t - k)$$
$$\psi_{j,k}(t) \coloneqq 2^{j/2}\psi(2^j t - k)$$

such that $\{\phi_{j,k}(t)\}_{k\in\mathbb{Z}}$ and $\{\psi_{j,k}(t)\}_{k\in\mathbb{Z}}$ are orthonormal bases for V_j and the proposed W_j respectively. Then we define the Haar wavelet space W_j as follows.

$$W_j \coloneqq Span\{\psi_{j,k}(t)\}_{k\in\mathbb{Z}} \cap L^2(\mathbb{R})$$

This space contains all possible residual functions g_j for approximating f_{j+1} by f_j.

Approximating $f_{j+1}(t) \in V_{j+1}$ by $f_j(t) \in V_j$

Let $f_{j+1}(t) = \sum_{k\in\mathbb{Z}} a_k \phi_{j+1,k}(t)$, $\mathbf{h} = \left[\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right]$ and $\mathbf{g} = \left[\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right]$.

Then the projection $f_j(t) = proj_{V_j} f_{j+1}(t)$ is $f_j(t) = \sum_{k\in\mathbb{Z}} \langle f_{j+1}(t), \phi_{j,k}(t) \rangle \phi_{j,k}(t)$, so

$f_j(t) = \sum_{k\in\mathbb{Z}} b_k \phi_{j,k}(t)$, where $b_k = \frac{\sqrt{2}}{2}(a_{2k} + a_{2k+1}) = \mathbf{h} \cdot [a_{2k}, a_{2k+1}]$.

Also, $g_j(t) = f_{j+1}(t) - f_j(t) = \sum_{k\in\mathbb{Z}} c_k \psi_{j,k}(t)$,

where $c_k = \frac{\sqrt{2}}{2}(a_{2k} - a_{2k+1}) = \mathbf{g} \cdot [a_{2k}, a_{2k+1}]$.

We call the decomposition from $f_{j+1}(t)$ to $f_j(t)$ and $g_j(t)$ discrete Haar wavelet transformation.

Reconstruction of $f_{j+1}(t)$ from $f_j(t)$ and $g_j(t)$
Reversing the process of decomposition, we can recover $f_{j+1}(t)$ by

$$a_{2k} = \frac{\sqrt{2}}{2}(b_k + c_k) = \mathbf{h} \cdot [b_k, c_k]$$
$$a_{2k+1} = \frac{\sqrt{2}}{2}(b_k - c_k) = \mathbf{g} \cdot [b_k, c_k]$$

This is called the inverse discrete Haar wavelet transformation.
Image compression algorithms use decomposition for multiple times to encode, use reconstruction iteratively to decode. Genius!

Implementation

Implementation in C++ is attempted, "work in progress" code can be found in the GitHub link provided.

Haar Transform in common sense

For pixel matrix M, we slide a 2×2 window through M.

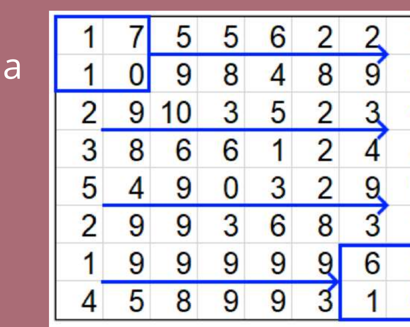
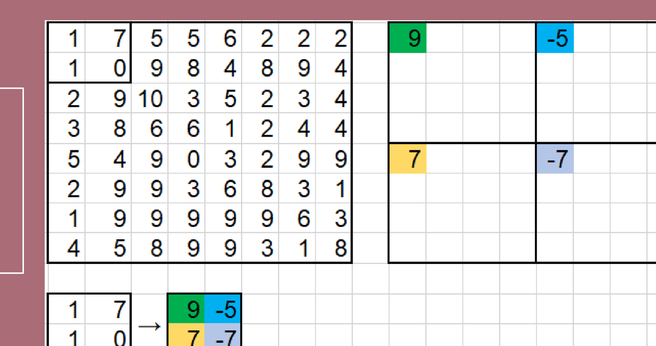


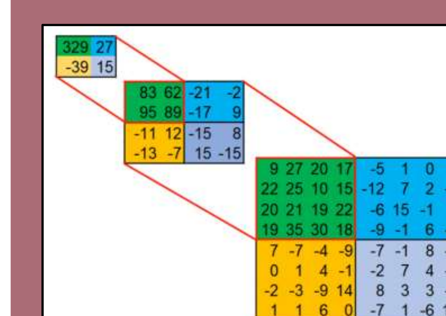
Figure: Sliding Window in M

Next, we can perform a "Pseudo Haar Transform" to the window A.

$$H = W^T A W$$
$$= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$
$$= \begin{bmatrix} a+b+c+d & a+c-b-d \\ a+b-c-d & a+d-b-c \end{bmatrix}$$
$$= \begin{bmatrix} \text{Sum} & \text{Horizontal Difference} \\ \text{Vertical Difference} & \text{Diagonal Difference} \end{bmatrix}$$

And store the data (as int) this way!





Such Process can be done repeatedly to have further performance!

Finally, we would apply Huffman Coding, to compress the result.

Acknowledgements

Special thanks to the following resources.

- GraphFree https://graphfree.com/
- MakeSigns https://www.makesigns.com/SciPosters_Templates.aspx
- *Wavelet Theory: An Elementary Approach with Application* https://doi.org/10.1002/9781118165652
- Haar wavelet based approach for image compression https://www.slideshare.net/veerendrabrrevanna/haar-wavelet-based-approach-for-image-compression
- Application of wavelet transform in spectrum sensing for cognitive radio: A survey https://www.semanticscholar.org/paper/Application-of-wavelet-transform-in-spectrum-for-A-Dibal-Onwuka/830336ea9a28ec5e0055eec6e3fad30526bb83cb

Analysis

The efficiency of the implemented compression algorithm is as below.

(Original Image: https://chengroupcuhk.org/research/)

Figures: Haar Transformed Images of CUHK Science Building (Left: 1 transform; Right: 2 transforms)

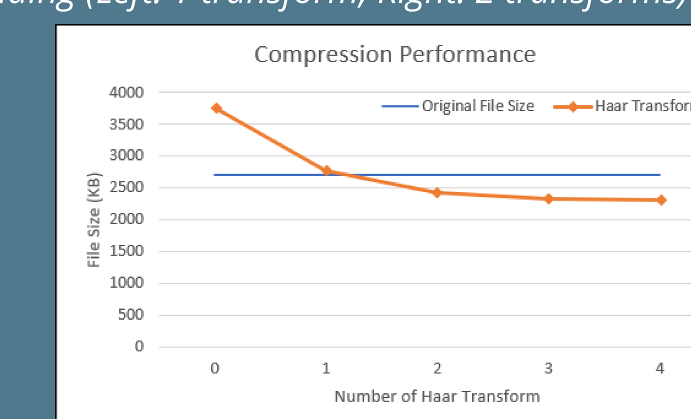


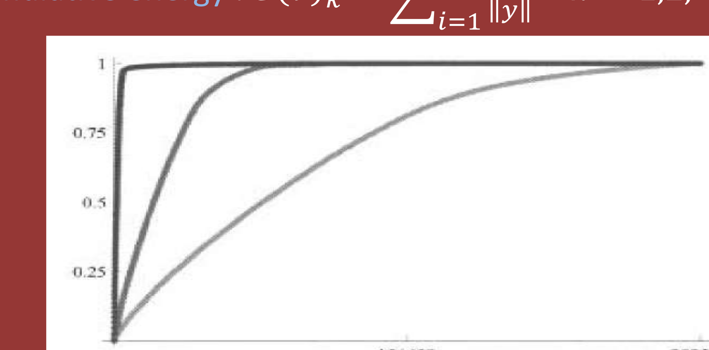
Figure: Compression Performance of the above image (without compression, the data is stored as `32-bit int`, therefore larger than the original file)

Best compression ratio (4 transforms)
= **1 : 0.854128**

Mechanics Behind Compression

To determine the effectiveness of discrete wavelet transformations, we introduce two measures.
Cumulative energy indicates which components might be important contributors to a signal.

Cumulative energy : $C(v)_k = \sum_{i=1}^{k} \frac{y_i^2}{\|y\|} \quad k = 1,2,\cdots N$



(The cumulative energy of the original image (light gray), one iteration (dark gray), and three iterations (black) of the discrete Haar wavelet transformation.)

Entropy represent the amount of information on average held by each unit of measure.

Entropy : $Ent(v) = \sum_{i=1}^{k} p(a_i) \log_2\left(\frac{1}{p(a_i)}\right)$

v=$[1,2,3,4,5,6,7,8]^T$ → HWT → $y = \left[\frac{3\sqrt{2}}{2}, \frac{7\sqrt{2}}{2}, \frac{11\sqrt{2}}{2}, \frac{15\sqrt{2}}{2}, \frac{-\sqrt{2}}{2}, \frac{-\sqrt{2}}{2}, \frac{-\sqrt{2}}{2}, \frac{-\sqrt{2}}{2}\right]^T$

Ent(v)=3 Ent(y)=2

The result y has a constant value in the detail portion which allows the entropy of y to be one less than that of v.
Transformations that can convert large blocks of elements to zero are the best for encoding methods.

Haar Wavelet Transform: Advantages and Disadvantages

Advantages

- Conceptually simple: Easy to implement and lower level of understanding is required.
- Fast computation: No need for multiplication. It requires only addition and there are many zero elements in the Haar matrix.
- Memory-efficient: Can be calculated in place without a temporary array.
- Lossless: Without the edge effects that are a problem with other wavelet transforms.

Disadvantages

- Shift sensitive: A shift in input image causes unpredictable changes in output.
- Lack of phase information: No description of the amplitude or local behaviour of an image.