THE DISCRETE HAAR WAVELET TRANSFORMATION

Patrick J. Van Fleet

Center for Applied Mathematics
University of St. Thomas
St. Paul, MN USA

Joint Mathematical Meetings, 7 & 9 January 2008

- Suppose you are given N values

$$\mathbf{x} = (x_1, x_2, \ldots, x_N)$$

  where N is even.
- Your task: Send an approximation $\mathbf{s}$ (a list of numbers) of this data via the internet to a colleague.
- In order to reduce transfer time, the length of your approximation must be $N/2$.
- How do you suggest we do it?

UNIVERSITY *of* ST. THOMAS

- Suppose you are given N values

$$\mathbf{x} = (x_1, x_2, \ldots, x_N)$$

where N is even.

- Your task: Send an approximation $\mathbf{s}$ (a list of numbers) of this data via the internet to a colleague.
- In order to reduce transfer time, the length of your approximation must be $N/2$.
- How do you suggest we do it?

- Suppose you are given N values

$$\mathbf{x} = (x_1, x_2, \ldots, x_N)$$

  where N is even.
- Your task: Send an approximation $\mathbf{s}$ (a list of numbers) of this data via the internet to a colleague.
- In order to reduce transfer time, the length of your approximation must be $N/2$.
- How do you suggest we do it?

- Suppose you are given N values

$$\mathbf{x} = (x_1, x_2, \ldots, x_N)$$

where N is even.

- Your task: Send an approximation $\mathbf{s}$ (a list of numbers) of this data via the internet to a colleague.
- In order to reduce transfer time, the length of your approximation must be $N/2$.
- How do you suggest we do it?

- One solution is to pair-wise average the numbers:

$$s_k = \frac{x_{2k-1} + x_{2k}}{2}, \qquad k = 1, \ldots, N/2$$

- For example:

$$\mathbf{x} = (6, 12, 15, 15, 14, 12, 120, 116) \rightarrow \mathbf{s} = (9, 15, 13, 118)$$

UNIVERSITY of ST. THOMAS

- One solution is to pair-wise average the numbers:

$$s_k = \frac{x_{2k-1} + x_{2k}}{2}, \qquad k = 1, \ldots, N/2$$

- For example:

$$\mathbf{x} = (6, 12, 15, 15, 14, 12, 120, 116) \rightarrow \mathbf{s} = (9, 15, 13, 118)$$

- Suppose now you were allowed to send extra data in addition to the pair-wise averages list **s**.
- The idea is to send a second list of data **d** so that the original list **x** can be recovered from **s** and **d**.
- How would you do it?

- Suppose now you were allowed to send extra data in addition to the pair-wise averages list **s**.
- The idea is to send a second list of data **d** so that the original list **x** can be recovered from **s** and **d**.
- How would you do it?

- Suppose now you were allowed to send extra data in addition to the pair-wise averages list **s**.
- The idea is to send a second list of data **d** so that the original list **x** can be recovered from **s** and **d**.
- How would you do it?

- Suppose now you were allowed to send extra data in addition to the pair-wise averages list **s**.
- The idea is to send a second list of data **d** so that the original list **x** can be recovered from **s** and **d**.
- How would you do it?

- There are a couple of choices for d_k (called directed distances):
- We could set

$$d_k = \frac{x_{2k-1} - x_{2k}}{2}, \qquad k = 1, \ldots, N/2$$

- or

$$d_k = \frac{x_{2k} - x_{2k-1}}{2}, \qquad k = 1, \ldots, N/2$$

- We will use the second formula.

$$\boxed{d_k = \frac{x_{2k} - x_{2k-1}}{2}, \qquad k = 1, \ldots, N/2}$$

University of St. Thomas

- There are a couple of choices for d_k (called directed distances):
- We could set

$$d_k = \frac{x_{2k-1} - x_{2k}}{2}, \qquad k = 1, \ldots, N/2$$

- or

$$d_k = \frac{x_{2k} - x_{2k-1}}{2}, \qquad k = 1, \ldots, N/2$$

- We will use the second formula.

$$\boxed{d_k = \frac{x_{2k} - x_{2k-1}}{2}, \qquad k = 1, \ldots, N/2}$$

University of St. Thomas

- There are a couple of choices for d_k (called directed distances):
- We could set

$$d_k = \frac{x_{2k-1} - x_{2k}}{2}, \qquad k = 1, \ldots, N/2$$

- or

$$d_k = \frac{x_{2k} - x_{2k-1}}{2}, \qquad k = 1, \ldots, N/2$$

- We will use the second formula.

$$\boxed{d_k = \frac{x_{2k} - x_{2k-1}}{2}, \qquad k = 1, \ldots, N/2}$$

University of St. Thomas

- There are a couple of choices for d_k (called directed distances):
- We could set

$$d_k = \frac{x_{2k-1} - x_{2k}}{2}, \qquad k = 1, \ldots, N/2$$

- or

$$d_k = \frac{x_{2k} - x_{2k-1}}{2}, \qquad k = 1, \ldots, N/2$$

- We will use the second formula.

$$\boxed{d_k = \frac{x_{2k} - x_{2k-1}}{2}, \qquad k = 1, \ldots, N/2}$$

- The process is invertible since

$$s_k + d_k = \frac{x_{2k-1} + x_{2k}}{2} + \frac{x_{2k} - x_{2k-1}}{2} = x_{2k}$$

and

$$s_k - d_k = \frac{x_{2k-1} + x_{2k}}{2} - \frac{x_{2k} - x_{2k-1}}{2} = x_{2k-1}$$

- So we map $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ to $(\mathbf{s} \,|\, \mathbf{d}) = (s_1, \ldots, s_{N/2} \,|\, d_1, \ldots, d_{N/2})$.
- Using our example values we have

$$(6, 12, 15, 15, 14, 12, 120, 116) \to (9, 15, 13, 118 \,|\, 3, 0, -1, -2)$$

- Why might people prefer the data in this form?

UNIVERSITY of ST. THOMAS

- The process is invertible since

$$s_k + d_k = \frac{x_{2k-1} + x_{2k}}{2} + \frac{x_{2k} - x_{2k-1}}{2} = x_{2k}$$

  and

$$s_k - d_k = \frac{x_{2k-1} + x_{2k}}{2} - \frac{x_{2k} - x_{2k-1}}{2} = x_{2k-1}$$

- So we map $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ to $(\mathbf{s} \,|\, \mathbf{d}) = (s_1, \ldots, s_{N/2} \,|\, d_1, \ldots, d_{N/2})$.
- Using our example values we have

$$(6, 12, 15, 15, 14, 12, 120, 116) \rightarrow (9, 15, 13, 118 \,|\, 3, 0, -1, -2)$$

- Why might people prefer the data in this form?

- The process is invertible since

$$s_k + d_k = \frac{x_{2k-1} + x_{2k}}{2} + \frac{x_{2k} - x_{2k-1}}{2} = x_{2k}$$

  and

$$s_k - d_k = \frac{x_{2k-1} + x_{2k}}{2} - \frac{x_{2k} - x_{2k-1}}{2} = x_{2k-1}$$

- So we map $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ to $(\mathbf{s} \,|\, \mathbf{d}) = (s_1, \ldots, s_{N/2} \,|\, d_1, \ldots, d_{N/2})$.
- Using our example values we have

$$(6, 12, 15, 15, 14, 12, 120, 116) \to (9, 15, 13, 118 \,|\, 3, 0, -1, -2)$$

- Why might people prefer the data in this form?

- The process is invertible since

$$s_k + d_k = \frac{x_{2k-1} + x_{2k}}{2} + \frac{x_{2k} - x_{2k-1}}{2} = x_{2k}$$

  and

$$s_k - d_k = \frac{x_{2k-1} + x_{2k}}{2} - \frac{x_{2k} - x_{2k-1}}{2} = x_{2k-1}$$

- So we map $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ to $(\mathbf{s} \,|\, \mathbf{d}) = (s_1, \ldots, s_{N/2} \,|\, d_1, \ldots, d_{N/2})$.
- Using our example values we have

$$(6, 12, 15, 15, 14, 12, 120, 116) \rightarrow (9, 15, 13, 118 \,|\, 3, 0, -1, -2)$$

- Why might people prefer the data in this form?

- We can identify large changes in the the differences portion **d** of the transform.
- It is easier to quantize the data in this form.
- The transform concentrates the signal's energy in fewer values.
- And the obvious answer: less digits!!
- We will talk about the top three bullets in due time.

- We can identify large changes in the the differences portion **d** of the transform.
- It is easier to quantize the data in this form.
- The transform concentrates the signal's energy in fewer values.
- And the obvious answer: less digits!!
- We will talk about the top three bullets in due time.

- We can identify large changes in the the differences portion **d** of the transform.
- It is easier to quantize the data in this form.
- The transform concentrates the signal's energy in fewer values.
- And the obvious answer: less digits!!
- We will talk about the top three bullets in due time.

University of St. Thomas

- We can identify large changes in the the differences portion **d** of the transform.
- It is easier to quantize the data in this form.
- The transform concentrates the signal's energy in fewer values.
- And the obvious answer: less digits!!
- We will talk about the top three bullets in due time.

UNIVERSITY of ST. THOMAS

- We can identify large changes in the the differences portion **d** of the transform.
- It is easier to quantize the data in this form.
- The transform concentrates the signal's energy in fewer values.
- And the obvious answer: less digits!!
- We will talk about the top three bullets in due time.

- The transformation

$$\mathbf{x} = (x_1, \ldots, x_N) \to (\mathbf{s} \,|\, \mathbf{d}) = (s_1, \ldots, s_{N/2} \,|\, d_1, \ldots, d_{N/2})$$

  is called the Discrete Haar Wavelet Transformation.
- What does the transform look like as a matrix?

- The transformation

$$\mathbf{x} = (x_1, \ldots, x_N) \to (\mathbf{s} \mid \mathbf{d}) = (s_1, \ldots, s_{N/2} \mid d_1, \ldots, d_{N/2})$$

  is called the Discrete Haar Wavelet Transformation.
- What does the transform look like as a matrix?

UNIVERSITY of ST. THOMAS

Consider applying the transform to an 8-vector. What is the matrix that works?

$$\begin{bmatrix} & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_1 + x_2 \\ x_3 + x_4 \\ x_5 + x_6 \\ x_7 + x_8 \\ \hline x_2 - x_1 \\ x_4 - x_3 \\ x_6 - x_5 \\ x_8 - x_7 \end{bmatrix}$$

Consider applying the transform to an 8-vector. What is the matrix that works?

$$\left[\begin{array}{cccccccc}
\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\
\hline
-\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2}
\end{array}\right] \cdot \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{array}\right] = \frac{1}{2}\left[\begin{array}{c} x_1 + x_2 \\ x_3 + x_4 \\ x_5 + x_6 \\ x_7 + x_8 \\ \hline x_2 - x_1 \\ x_4 - x_3 \\ x_6 - x_5 \\ x_8 - x_7 \end{array}\right]$$

We will denote the transform matrix by W_8.

University of St. Thomas

What about W_8^{-1}? That is, what matrix solves

$$\left[\begin{array}{cccccccc} & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \end{array}\right] \cdot \left(\frac{1}{2} \left[\begin{array}{c} x_1 + x_2 \\ x_3 + x_4 \\ x_5 + x_6 \\ x_7 + x_8 \\ \hline x_2 - x_1 \\ x_4 - x_3 \\ x_6 - x_5 \\ x_8 - x_7 \end{array}\right] \right) = \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{array}\right]$$

What about W_8^{-1}? That is, what matrix solves

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \cdot \left(\frac{1}{2} \begin{bmatrix} x_1 + x_2 \\ x_3 + x_4 \\ x_5 + x_6 \\ x_7 + x_8 \\ \hline x_2 - x_1 \\ x_4 - x_3 \\ x_6 - x_5 \\ x_8 - x_7 \end{bmatrix} \right) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$$

- Learning how to code the HWT and its inverse provides a good review of linear algebra.
- We'll use $N = 8$ as an example. Let

$$W_8 = \left[\begin{array}{cccccccc} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \hline -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} \end{array}\right] = \left[\begin{array}{c} H \\ \hline G \end{array}\right]$$

UNIVERSITY of ST. THOMAS

- Learning how to code the HWT and its inverse provides a good review of linear algebra.
- We'll use $N = 8$ as an example. Let

$$W_8 = \left[\begin{array}{cccccccc} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \hline -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} \end{array}\right] = \left[\begin{array}{c} H \\ \hline G \end{array}\right]$$

University of St. Thomas

- Then

$$W_8\mathbf{x} = \begin{bmatrix} H \\ - \\ G \end{bmatrix} \mathbf{x} = \begin{bmatrix} H\mathbf{x} \\ - \\ G\mathbf{x} \end{bmatrix}$$

- Let's look at

$$H \cdot \mathbf{x} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$$

University of St. Thomas

- Then

$$W_8\mathbf{x} = \begin{bmatrix} H \\ - \\ G \end{bmatrix} \mathbf{x} = \begin{bmatrix} H\mathbf{x} \\ - \\ G\mathbf{x} \end{bmatrix}$$

- Let's look at

$$H \cdot \mathbf{x} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$$

We have

$$H \cdot \mathbf{x} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix}$$

$$= \frac{1}{2} \begin{bmatrix} x_1 + x_2 \\ x_3 + x_4 \\ x_5 + x_6 \\ x_7 + x_8 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \\ x_5 & x_6 \\ x_7 & x_8 \end{bmatrix} \cdot \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$$

- In a similar manner, we have

$$G \cdot \mathbf{x} = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \\ x_5 & x_6 \\ x_7 & x_8 \end{bmatrix} \cdot \begin{bmatrix} -1/2 \\ 1/2 \end{bmatrix}$$

- Thus to code $W_N \cdot \mathbf{x}$, we
  - Partition the input $\mathbf{x}$ into

$$X = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \\ & \vdots \\ x_{N-1} & x_N \end{bmatrix}$$

  - Compute $\mathbf{s} = X \cdot \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$, $\mathbf{d} = X \cdot \begin{bmatrix} -1/2 \\ 1/2 \end{bmatrix}$
  - Return $[\mathbf{s} \mid \mathbf{d}]$

- In a similar manner, we have

$$G \cdot \mathbf{x} = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \\ x_5 & x_6 \\ x_7 & x_8 \end{bmatrix} \cdot \begin{bmatrix} -1/2 \\ 1/2 \end{bmatrix}$$

- Thus to code $W_N \cdot \mathbf{x}$, we
  - Partition the input $\mathbf{x}$ into

$$X = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \\ & \vdots \\ x_{N-1} & x_N \end{bmatrix}$$

  - Compute $\mathbf{s} = X \cdot \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$, $\mathbf{d} = X \cdot \begin{bmatrix} -1/2 \\ 1/2 \end{bmatrix}$
  - Return $[\mathbf{s} \,|\, \mathbf{d}]$

- In a similar manner, we have

$$G \cdot \mathbf{x} = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \\ x_5 & x_6 \\ x_7 & x_8 \end{bmatrix} \cdot \begin{bmatrix} -1/2 \\ 1/2 \end{bmatrix}$$

- Thus to code $W_N \cdot \mathbf{x}$, we
  - Partition the input $\mathbf{x}$ into

$$X = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \\ \vdots & \\ x_{N-1} & x_N \end{bmatrix}$$

  - Compute $\mathbf{s} = X \cdot \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$, $\mathbf{d} = X \cdot \begin{bmatrix} -1/2 \\ 1/2 \end{bmatrix}$
  - Return $[\mathbf{s} \,|\, \mathbf{d}]$

UNIVERSITY of ST. THOMAS

- In a similar manner, we have

$$G \cdot \mathbf{x} = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \\ x_5 & x_6 \\ x_7 & x_8 \end{bmatrix} \cdot \begin{bmatrix} -1/2 \\ 1/2 \end{bmatrix}$$

- Thus to code $W_N \cdot \mathbf{x}$, we
  - Partition the input $\mathbf{x}$ into

$$X = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \\ & \vdots \\ x_{N-1} & x_N \end{bmatrix}$$

  - Compute $\mathbf{s} = X \cdot \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$, $\mathbf{d} = X \cdot \begin{bmatrix} -1/2 \\ 1/2 \end{bmatrix}$
  - Return $[\mathbf{s} \mid \mathbf{d}]$

University of St. Thomas

- In a similar manner, we have

$$G \cdot \mathbf{x} = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \\ x_5 & x_6 \\ x_7 & x_8 \end{bmatrix} \cdot \begin{bmatrix} -1/2 \\ 1/2 \end{bmatrix}$$

- Thus to code $W_N \cdot \mathbf{x}$, we
  - Partition the input $\mathbf{x}$ into

$$X = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \\ & \vdots \\ x_{N-1} & x_N \end{bmatrix}$$

  - Compute $\mathbf{s} = X \cdot \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$, $\mathbf{d} = X \cdot \begin{bmatrix} -1/2 \\ 1/2 \end{bmatrix}$
  - Return $[\mathbf{s} \,|\, \mathbf{d}]$

University of St. Thomas

Here is a simple Mathematica module to do the job:

```
HWT1D[x_]:=Module[{X,s,d,y},
     X=Partition[x,2,2];
     s=X.{1/2,1/2};
     d=X.{-1/2,1/2};
     y=Join[s,d];
     Return[y];
];
```

Here is a simple Matlab function:

```
function y=HWT1D(x)
    N=length(x)
    X=reshape(x,2,N/2);
    s=X. [.5; .5 ];
    d=X. [-.5; .5 ];
    y=[s; d];
```

The coding for the inverse is similar but with a different twist at the end. We need an algorithm for computing

$$W_8^{-1} \cdot \mathbf{y} = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} y_1 - y_5 \\ y_1 + y_5 \\ y_2 - y_6 \\ y_2 + y_6 \\ y_3 - y_7 \\ y_3 + y_7 \\ y_4 - y_8 \\ y_4 + y_8 \end{bmatrix}$$

- We could define the matrix

$$Y = \begin{bmatrix} y_1 & y_5 \\ y_2 & y_6 \\ y_3 & y_7 \\ y_4 & y_8 \end{bmatrix}$$

- and then compute

$$\mathbf{a} = Y \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} y_1 - y_5 \\ y_2 - y_6 \\ y_3 - y_7 \\ y_4 - y_8 \end{bmatrix}, \qquad \mathbf{b} = Y \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} y_1 + y_5 \\ y_2 + y_6 \\ y_3 + y_7 \\ y_4 + y_8 \end{bmatrix}$$

- We return

$$(a_1, b_1, a_2, b_2, a_3, b_3, a_4, b_4)$$

- We could define the matrix

$$Y = \begin{bmatrix} y_1 & y_5 \\ y_2 & y_6 \\ y_3 & y_7 \\ y_4 & y_8 \end{bmatrix}$$

- and then compute

$$\mathbf{a} = Y \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} y_1 - y_5 \\ y_2 - y_6 \\ y_3 - y_7 \\ y_4 - y_8 \end{bmatrix}, \qquad \mathbf{b} = Y \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} y_1 + y_5 \\ y_2 + y_6 \\ y_3 + y_7 \\ y_4 + y_8 \end{bmatrix}$$

- We return

$$(a_1, b_1, a_2, b_2, a_3, b_3, a_4, b_4)$$

- We could define the matrix

$$Y = \begin{bmatrix} y_1 & y_5 \\ y_2 & y_6 \\ y_3 & y_7 \\ y_4 & y_8 \end{bmatrix}$$

- and then compute

$$\mathbf{a} = Y \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} y_1 - y_5 \\ y_2 - y_6 \\ y_3 - y_7 \\ y_4 - y_8 \end{bmatrix}, \qquad \mathbf{b} = Y \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} y_1 + y_5 \\ y_2 + y_6 \\ y_3 + y_7 \\ y_4 + y_8 \end{bmatrix}$$

- We return

$$(a_1, b_1, a_2, b_2, a_3, b_3, a_4, b_4)$$

UNIVERSITY of ST. THOMAS

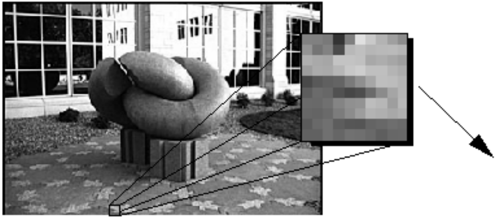
Here is a simple Mathematica module to do the job:

```
IHWT1D[y_]:=Module[{Y,a,b,x},
     Y=Transpose[Partition[y,Length[y]/2]];
     a=Y.{1,-1};
     b=X.{1,1};
     x=Transpose[{a, b}];
     Return[Flatten[x]];
];
```

Let's have a look at the Mathematica notebook

`HaarTransform1D.nb`

An *8-bit* digital image can be viewed as a matrix whose entries (known as *pixels*) range from 0 (black) to 255 (white).



$$\begin{bmatrix} 129 & 128 & 121 & 51 & 127 & 224 & 201 & 179 & 159 & 140 \\ 148 & 116 & 130 & 75 & 184 & 191 & 182 & 185 & 186 & 180 \\ 175 & 169 & 166 & 195 & 195 & 192 & 168 & 173 & 166 & 158 \\ 157 & 171 & 169 & 182 & 199 & 205 & 191 & 191 & 180 & 172 \\ 73 & 89 & 96 & 100 & 122 & 143 & 166 & 190 & 188 & 180 \\ 93 & 107 & 103 & 81 & 70 & 77 & 106 & 139 & 165 & 181 \\ 106 & 105 & 112 & 132 & 144 & 147 & 189 & 183 & 158 & 184 \\ 102 & 100 & 106 & 124 & 140 & 157 & 179 & 175 & 168 & 175 \\ 91 & 105 & 112 & 93 & 86 & 85 & 100 & 104 & 110 & 106 \\ 97 & 97 & 112 & 102 & 113 & 111 & 105 & 94 & 103 & 104 \end{bmatrix}$$

University of St. Thomas

Consider the 480×640 image (call it A)

If $\mathbf{a}^1, \ldots, \mathbf{a}^{640}$ are the columns of A, then computing $W_{640}A$ is the same as applying the HWT to each column of A:

$$W_{640}A = (W_{640} \cdot \mathbf{a}^1, \ldots, W_{640} \cdot \mathbf{a}^{640})$$

Graphically, we have

- $C = W_{640}A$ processes the columns of A.
- How would we process the rows of C?
- We compute $CW_{480}^T = W_{640}AW_{480}^T$.
- The two-dimensional Haar transform of $M \times N$ matrix A is

$$\boxed{B = W_N A W_M^T}$$

- $C = W_{640}A$ processes the columns of A.
- How would we process the rows of C?
- We compute $CW_{480}^T = W_{640}AW_{480}^T$.
- The two-dimensional Haar transform of $M \times N$ matrix A is

$$\boxed{B = W_N A W_M^T}$$

- $C = W_{640}A$ processes the columns of A.
- How would we process the rows of C?
- We compute $CW_{480}^T = W_{640}AW_{480}^T$.
- The two-dimensional Haar transform of $M \times N$ matrix A is

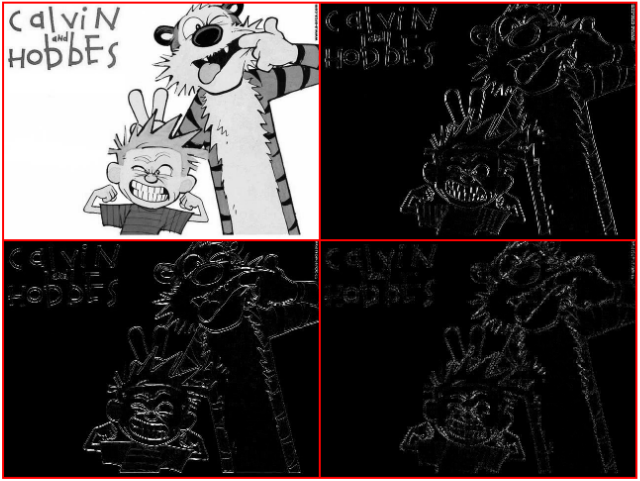
$$\boxed{B = W_N A W_M^T}$$

UNIVERSITY of ST. THOMAS

- $C = W_{640}A$ processes the columns of A.
- How would we process the rows of C?
- We compute $CW_{480}^T = W_{640}AW_{480}^T$.
- The two-dimensional Haar transform of $M \times N$ matrix A is

$$\boxed{B = W_N A W_M^T}$$

Graphically, we have



- Can we interpret what the transformation does to the image?
- Suppose A is the 4×4 matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

- Partitioning $W_4 = \begin{bmatrix} H \\ - \\ G \end{bmatrix}$, we have

UNIVERSITY of ST. THOMAS

- Can we interpret what the transformation does to the image?
- Suppose A is the 4×4 matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

- Partitioning $W_4 = \begin{bmatrix} H \\ - \\ G \end{bmatrix}$, we have

- Can we interpret what the transformation does to the image?
- Suppose A is the 4×4 matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

- Partitioning $W_4 = \begin{bmatrix} H \\ - \\ G \end{bmatrix}$, we have

University of St. Thomas

$$
\begin{aligned}
W_4 A W_4^T &= \left[\begin{array}{c} H \\ - \\ G \end{array}\right] A \left[\begin{array}{c|c} H^T & G^T \end{array}\right] \\
&= \left[\begin{array}{c} HA \\ - \\ GA \end{array}\right] \left[\begin{array}{c|c} H^T & G^T \end{array}\right] \\
&= \left[\begin{array}{c|c} HAH^T & HAG^T \\ \hline GAH^T & GAG^T \end{array}\right]
\end{aligned}
$$

Let's look at each 2×2 block individually:

- $HAH^T = \frac{1}{4}\left[\begin{array}{c|c} a_{11}+a_{12}+a_{21}+a_{22} & a_{13}+a_{14}+a_{23}+a_{24} \\ \hline a_{31}+a_{32}+a_{41}+a_{42} & a_{33}+a_{34}+a_{43}+a_{44} \end{array}\right]$
- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- Then the (i, j) element of HAH^T is simply the average of the elements in A_{ij}!
- So HAH^T is an approximation or blur of the original image. We will denote HAH^T as $\mathcal{B}$.

UNIVERSITY *of* ST. THOMAS

- $HAH^T = \frac{1}{4}\left[\begin{array}{c|c} a_{11}+a_{12}+a_{21}+a_{22} & a_{13}+a_{14}+a_{23}+a_{24} \\ \hline a_{31}+a_{32}+a_{41}+a_{42} & a_{33}+a_{34}+a_{43}+a_{44} \end{array}\right]$
- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- Then the (i, j) element of HAH^T is simply the average of the elements in A_{ij}!
- So HAH^T is an approximation or blur of the original image. We will denote HAH^T as $\mathcal{B}$.

- $HAH^T = \frac{1}{4}\left[\begin{array}{c|c} a_{11}+a_{12}+a_{21}+a_{22} & a_{13}+a_{14}+a_{23}+a_{24} \\ \hline a_{31}+a_{32}+a_{41}+a_{42} & a_{33}+a_{34}+a_{43}+a_{44} \end{array}\right]$
- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- Then the (i, j) element of HAH^T is simply the average of the elements in A_{ij}!
- So HAH^T is an approximation or blur of the original image. We will denote HAH^T as $\mathcal{B}$.

UNIVERSITY of ST. THOMAS

- $HAH^T = \frac{1}{4}\left[\begin{array}{c|c} a_{11}+a_{12}+a_{21}+a_{22} & a_{13}+a_{14}+a_{23}+a_{24} \\ \hline a_{31}+a_{32}+a_{41}+a_{42} & a_{33}+a_{34}+a_{43}+a_{44} \end{array}\right]$
- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- Then the (i, j) element of HAH^T is simply the average of the elements in A_{ij}!
- So HAH^T is an approximation or blur of the original image. We will denote HAH^T as $\mathcal{B}$.

- The upper right hand corner is

$$HAG^T = \frac{1}{4}\left[\begin{array}{cc} (a_{12}+a_{22})-(a_{11}+a_{21}) & (a_{14}+a_{24})-(a_{13}+a_{23}) \\ (a_{32}+a_{42})-(a_{31}+a_{41}) & (a_{34}+a_{44})-(a_{33}+a_{43}) \end{array}\right]$$

- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- The (i,j) element of HAG^T can be viewed as differences along columns of A_{ij}.
- We will denote HAG^T as $\mathcal{V}$ (for vertical differences).

UNIVERSITY of ST. THOMAS

- The upper right hand corner is

$$HAG^T = \frac{1}{4}\begin{bmatrix} (a_{12}+a_{22})-(a_{11}+a_{21}) & (a_{14}+a_{24})-(a_{13}+a_{23}) \\ (a_{32}+a_{42})-(a_{31}+a_{41}) & (a_{34}+a_{44})-(a_{33}+a_{43}) \end{bmatrix}$$

- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- The (i,j) element of HAG^T can be viewed as differences along columns of A_{ij}.
- We will denote HAG^T as $\mathcal{V}$ (for vertical differences).

- The upper right hand corner is

$$HAG^T = \frac{1}{4}\begin{bmatrix} (a_{12}+a_{22})-(a_{11}+a_{21}) & (a_{14}+a_{24})-(a_{13}+a_{23}) \\ (a_{32}+a_{42})-(a_{31}+a_{41}) & (a_{34}+a_{44})-(a_{33}+a_{43}) \end{bmatrix}$$

- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- The (i, j) element of HAG^T can be viewed as differences along columns of A_{ij}.
- We will denote HAG^T as $\mathcal{V}$ (for vertical differences).

UNIVERSITY of ST. THOMAS

- The upper right hand corner is

$$HAG^T = \frac{1}{4}\begin{bmatrix} (a_{12}+a_{22})-(a_{11}+a_{21}) & (a_{14}+a_{24})-(a_{13}+a_{23}) \\ (a_{32}+a_{42})-(a_{31}+a_{41}) & (a_{34}+a_{44})-(a_{33}+a_{43}) \end{bmatrix}$$

- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- The (i,j) element of HAG^T can be viewed as differences along columns of A_{ij}.
- We will denote HAG^T as $\mathcal{V}$ (for vertical differences).

- The lower left hand corner is

$$GAH^T = \frac{1}{4}\left[\begin{array}{cc} (a_{21}+a_{22})-(a_{12}+a_{11}) & (a_{23}+a_{24})-(a_{13}+a_{14}) \\ (a_{31}+a_{32})-(a_{42}+a_{41}) & (a_{43}+a_{44})-(a_{33}+a_{34}) \end{array}\right]$$

- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- The (i,j) element of HAG^T can be viewed as differences along rows of A_{ij}.
- We will denote GAH^T as $\mathcal{H}$ (for horizontal differences).

- The lower left hand corner is

$$GAH^T = \frac{1}{4}\begin{bmatrix} (a_{21}+a_{22})-(a_{12}+a_{11}) & (a_{23}+a_{24})-(a_{13}+a_{14}) \\ (a_{31}+a_{32})-(a_{42}+a_{41}) & (a_{43}+a_{44})-(a_{33}+a_{34}) \end{bmatrix}$$

- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- The (i, j) element of HAG^T can be viewed as differences along rows of A_{ij}.
- We will denote GAH^T as $\mathcal{H}$ (for horizontal differences).

- The lower left hand corner is

$$GAH^T = \frac{1}{4}\begin{bmatrix} (a_{21}+a_{22})-(a_{12}+a_{11}) & (a_{23}+a_{24})-(a_{13}+a_{14}) \\ (a_{31}+a_{32})-(a_{42}+a_{41}) & (a_{43}+a_{44})-(a_{33}+a_{34}) \end{bmatrix}$$

- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- The (i,j) element of HAG^T can be viewed as differences along rows of A_{ij}.
- We will denote GAH^T as $\mathcal{H}$ (for horizontal differences).

- The lower left hand corner is

$$GAH^T = \frac{1}{4}\begin{bmatrix} (a_{21}+a_{22})-(a_{12}+a_{11}) & (a_{23}+a_{24})-(a_{13}+a_{14}) \\ (a_{31}+a_{32})-(a_{42}+a_{41}) & (a_{43}+a_{44})-(a_{33}+a_{34}) \end{bmatrix}$$

- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- The (i, j) element of HAG^T can be viewed as differences along rows of A_{ij}.
- We will denote GAH^T as $\mathcal{H}$ (for horizontal differences).

- The lower right hand corner is

$$GAG^T = \frac{1}{4}\begin{bmatrix} (a_{11}+a_{22})-(a_{12}+a_{21}) & (a_{13}+a_{24})-(a_{23}+a_{14}) \\ (a_{31}+a_{42})-(a_{32}+a_{41}) & (a_{33}+a_{44})-(a_{43}+a_{34}) \end{bmatrix}$$

- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- The (i,j) element of GAG^T can be viewed as differences along the diagonal of A_{ij}.
- We will denote GAG^T as $\mathcal{D}$ (for diagonal differences).

- The lower right hand corner is

$$GAG^T = \frac{1}{4}\begin{bmatrix} (a_{11}+a_{22})-(a_{12}+a_{21}) & (a_{13}+a_{24})-(a_{23}+a_{14}) \\ (a_{31}+a_{42})-(a_{32}+a_{41}) & (a_{33}+a_{44})-(a_{43}+a_{34}) \end{bmatrix}$$

- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- The (i,j) element of GAG^T can be viewed as differences along the diagonal of A_{ij}.
- We will denote GAG^T as $\mathcal{D}$ (for diagonal differences).

UNIVERSITY of ST. THOMAS

- The lower right hand corner is

$$GAG^T = \frac{1}{4}\left[\begin{array}{cc} (a_{11}+a_{22})-(a_{12}+a_{21}) & (a_{13}+a_{24})-(a_{23}+a_{14}) \\ (a_{31}+a_{42})-(a_{32}+a_{41}) & (a_{33}+a_{44})-(a_{43}+a_{34}) \end{array}\right]$$

- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- The (i, j) element of GAG^T can be viewed as differences along the diagonal of A_{ij}.
- We will denote GAG^T as $\mathcal{D}$ (for diagonal differences).

University of St. Thomas

- The lower right hand corner is

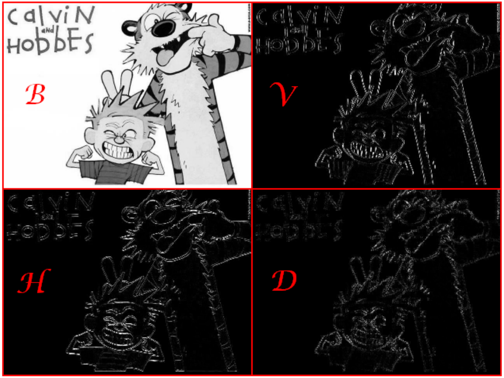
$$GAG^T = \frac{1}{4}\begin{bmatrix} (a_{11}+a_{22})-(a_{12}+a_{21}) & (a_{13}+a_{24})-(a_{23}+a_{14}) \\ (a_{31}+a_{42})-(a_{32}+a_{41}) & (a_{33}+a_{44})-(a_{43}+a_{34}) \end{bmatrix}$$

- Partition A in 2×2 blocks as

$$A = \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right]$$

- The (i,j) element of GAG^T can be viewed as differences along the diagonal of A_{ij}.
- We will denote GAG^T as $\mathcal{D}$ (for diagonal differences).

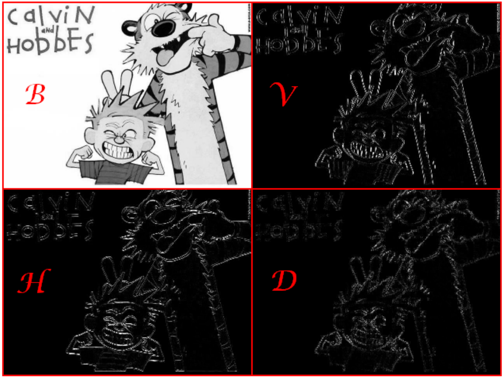
- So again the transform of our image is



- Once we have coded up the 2D HWT, I have the students write some Mathematica code to discover the parts of the transform.
- They can also think about how to build the inverse using the same code.

UNIVERSITY of ST. THOMAS

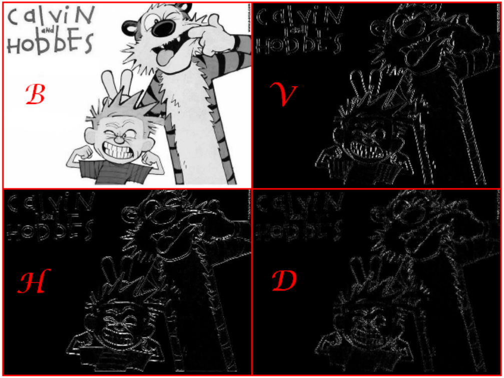
- So again the transform of our image is



- Once we have coded up the 2D HWT, I have the students write some Mathematica code to discover the parts of the transform.
- They can also think about how to build the inverse using the same code.

University of St. Thomas

- So again the transform of our image is



- Once we have coded up the 2D HWT, I have the students write some Mathematica code to discover the parts of the transform.
- They can also think about how to build the inverse using the same code.

- Writing code for the 2D Haar transform is easy and a good review of some basic properties from linear algebra.
- Given an $M \times N$ matrix A, we wish to compute

$$B = W_M A W_N^T$$

- We start by computing $C = W_M A$. This is easy - we simply apply `HWT1D` to each column of A.
- Mathematica is a "row-oriented" language so to perform $W_M A$, we must transpose A, apply `HWT1D` to A^T and then transpose back.
- Here is the Mathematica module:

```
LeftHaar[a_]:=Transpose[Map[HWT1D,Transpose[a]]];
```

- Writing code for the 2D Haar transform is easy and a good review of some basic properties from linear algebra.
- Given an $M \times N$ matrix A, we wish to compute

$$B = W_M A W_N^T$$

- We start by computing $C = W_M A$. This is easy - we simply apply `HWT1D` to each column of A.
- Mathematica is a "row-oriented" language so to perform $W_M A$, we must transpose A, apply `HWT1D` to A^T and then transpose back.
- Here is the Mathematica module:

```
LeftHaar[a_]:=Transpose[Map[HWT1D,Transpose[a]]];
```

UNIVERSITY of ST. THOMAS

- Writing code for the 2D Haar transform is easy and a good review of some basic properties from linear algebra.
- Given an $M \times N$ matrix A, we wish to compute

$$B = W_M A W_N^T$$

- We start by computing $C = W_M A$. This is easy - we simply apply `HWT1D` to each column of A.
- Mathematica is a "row-oriented" language so to perform $W_M A$, we must transpose A, apply `HWT1D` to A^T and then transpose back.
- Here is the Mathematica module:

```
LeftHaar[a_]:=Transpose[Map[HWT1D,Transpose[a]]];
```

UNIVERSITY *of* ST. THOMAS

- Writing code for the 2D Haar transform is easy and a good review of some basic properties from linear algebra.
- Given an $M \times N$ matrix A, we wish to compute

$$B = W_M A W_N^T$$

- We start by computing $C = W_M A$. This is easy - we simply apply `HWT1D` to each column of A.
- Mathematica is a "row-oriented" language so to perform $W_M A$, we must transpose A, apply `HWT1D` to A^T and then transpose back.
- Here is the Mathematica module:

```
LeftHaar[a_]:=Transpose[Map[HWT1D,Transpose[a]]];
```

UNIVERSITY of ST. THOMAS

- Writing code for the 2D Haar transform is easy and a good review of some basic properties from linear algebra.
- Given an $M \times N$ matrix A, we wish to compute

$$B = W_M A W_N^T$$

- We start by computing $C = W_M A$. This is easy - we simply apply `HWT1D` to each column of A.
- Mathematica is a "row-oriented" language so to perform $W_M A$, we must transpose A, apply `HWT1D` to A^T and then transpose back.
- Here is the Mathematica module:

```
LeftHaar[a_]:=Transpose[Map[HWT1D,Transpose[a]]];
```

UNIVERSITY of ST. THOMAS

- So now we have a module (`LeftHaar`) for computing $C = W_M A$.
- The final computation is $B = W_M A W_N^T = C W_N^T$.
- Rather than writing a routine for right multiplying by W_N^T, let's use some basic linear algebra:

$$B^T = (CW_N^T)^T = W_N C^T$$

- We already have a routine for left multiplying by W_N so we simply transpose C, apply `LeftHaar` to C^T and then transpose the result!
- We can even leave out the transpose steps:

```
HWT2D[a_] := Map[HWT1D, LeftHaar[a]];
```

UNIVERSITY of ST. THOMAS

- So now we have a module (`LeftHaar`) for computing $C = W_M A$.
- The final computation is $B = W_M A W_N^T = C W_N^T$.
- Rather than writing a routine for right multiplying by W_N^T, let's use some basic linear algebra:

$$B^T = (C W_N^T)^T = W_N C^T$$

- We already have a routine for left multiplying by W_N so we simply transpose C, apply `LeftHaar` to C^T and then transpose the result!
- We can even leave out the transpose steps:

```
HWT2D[a_] := Map[HWT1D, LeftHaar[a]];
```

UNIVERSITY of ST. THOMAS

- So now we have a module (`LeftHaar`) for computing $C = W_M A$.
- The final computation is $B = W_M A W_N^T = C W_N^T$.
- Rather than writing a routine for right multiplying by W_N^T, let's use some basic linear algebra:

$$B^T = (C W_N^T)^T = W_N C^T$$

- We already have a routine for left multiplying by W_N so we simply transpose C, apply `LeftHaar` to C^T and then transpose the result!
- We can even leave out the transpose steps:

```
HWT2D[a_] := Map[HWT1D, LeftHaar[a]];
```

UNIVERSITY of ST. THOMAS

- So now we have a module (`LeftHaar`) for computing $C = W_M A$.
- The final computation is $B = W_M A W_N^T = C W_N^T$.
- Rather than writing a routine for right multiplying by W_N^T, let's use some basic linear algebra:

$$B^T = (C W_N^T)^T = W_N C^T$$

- We already have a routine for left multiplying by W_N so we simply transpose C, apply `LeftHaar` to C^T and then transpose the result!
- We can even leave out the transpose steps:

```
HWT2D[a_] := Map[HWT1D, LeftHaar[a]];
```

University of St. Thomas

- So now we have a module (`LeftHaar`) for computing $C = W_M A$.
- The final computation is $B = W_M A W_N^T = C W_N^T$.
- Rather than writing a routine for right multiplying by W_N^T, let's use some basic linear algebra:

$$B^T = (C W_N^T)^T = W_N C^T$$

- We already have a routine for left multiplying by W_N so we simply transpose C, apply `LeftHaar` to C^T and then transpose the result!
- We can even leave out the transpose steps:

```
HWT2D[a_] := Map[HWT1D, LeftHaar[a]];
```

Let's have a look at the Mathematica notebook

`HaarTransform2D.nb`

- A quick application of the 2D Haar Transform is edge detection in digital images.
- The process is quite simple.
  - Compute i iterations of the HWT on A.
  - Replace the blur $\mathcal{B}$ by a zero matrix of the same size.
  - Compute i iterations of the inverse HWT to obtain the edges in the image.
- Let's have a look at the notebook

`HaarEdgeDetection.nb`

- A quick application of the 2D Haar Transform is edge detection in digital images.
- The process is quite simple.
  - Compute i iterations of the HWT on A.
  - Replace the blur $\mathcal{B}$ by a zero matrix of the same size.
  - Compute i iterations of the inverse HWT to obtain the edges in the image.
- Let's have a look at the notebook

`HaarEdgeDetection.nb`

- A quick application of the 2D Haar Transform is edge detection in digital images.
- The process is quite simple.
  - Compute i iterations of the HWT on A.
  - Replace the blur $\mathcal{B}$ by a zero matrix of the same size.
  - Compute i iterations of the inverse HWT to obtain the edges in the image.
- Let's have a look at the notebook

`HaarEdgeDetection.nb`

University of St. Thomas

- A quick application of the 2D Haar Transform is edge detection in digital images.
- The process is quite simple.
  - Compute i iterations of the HWT on A.
  - Replace the blur $\mathcal{B}$ by a zero matrix of the same size.
  - Compute i iterations of the inverse HWT to obtain the edges in the image.
- Let's have a look at the notebook

`HaarEdgeDetection.nb`

- A quick application of the 2D Haar Transform is edge detection in digital images.
- The process is quite simple.
  - Compute i iterations of the HWT on A.
  - Replace the blur $\mathcal{B}$ by a zero matrix of the same size.
  - Compute i iterations of the inverse HWT to obtain the edges in the image.
- Let's have a look at the notebook

`HaarEdgeDetection.nb`

- A quick application of the 2D Haar Transform is edge detection in digital images.
- The process is quite simple.
  - Compute i iterations of the HWT on A.
  - Replace the blur $\mathcal{B}$ by a zero matrix of the same size.
  - Compute i iterations of the inverse HWT to obtain the edges in the image.
- Let's have a look at the notebook

`HaarEdgeDetection.nb`

- The basic (wavelet-based) image compression works as follows:
  - Perform i iterations of the HWT on A.
  - Quantize the transformed image.
  - Encode the quantized data.
  - Transmit/store.
- The quantization step makes the compression lossy.

- The basic (wavelet-based) image compression works as follows:
  - Perform i iterations of the HWT on A.
  - Quantize the transformed image.
  - Encode the quantized data.
  - Transmit/store.
- The quantization step makes the compression lossy.

- The basic (wavelet-based) image compression works as follows:
  - Perform i iterations of the HWT on A.
  - Quantize the transformed image.
  - Encode the quantized data.
  - Transmit/store.
- The quantization step makes the compression lossy.

- The basic (wavelet-based) image compression works as follows:
  - Perform i iterations of the HWT on A.
  - Quantize the transformed image.
  - Encode the quantized data.
  - Transmit/store.
- The quantization step makes the compression lossy.

- The basic (wavelet-based) image compression works as follows:
  - Perform i iterations of the HWT on A.
  - Quantize the transformed image.
  - Encode the quantized data.
  - Transmit/store.
- The quantization step makes the compression lossy.

- The basic (wavelet-based) image compression works as follows:
  - Perform i iterations of the HWT on A.
  - Quantize the transformed image.
  - Encode the quantized data.
  - Transmit/store.
- The quantization step makes the compression lossy.

- Cumulative Energy is a simple vector-valued function that gives information about the concentration of energy in a vector.
- Let $\mathbf{v} = (v_1, \ldots, v_N) \in \mathbb{R}^N$.
  - Take the absolute value of each component of $\mathbf{v}$ and sort from largest to smallest. Call this new vector $\mathbf{y}$.
  - Note $\|\mathbf{v}\| = \|\mathbf{y}\|$.
  - Define

$$\boxed{CE_k = \sum_{j=1}^{k} \frac{y_k^2}{\|\mathbf{y}\|^2}}$$

- Cumulative Energy is a simple vector-valued function that gives information about the concentration of energy in a vector.
- Let $\mathbf{v} = (v_1, \ldots, v_N) \in \mathbb{R}^N$.
  - Take the absolute value of each component of $\mathbf{v}$ and sort from largest to smallest. Call this new vector $\mathbf{y}$.
  - Note $\|\mathbf{v}\| = \|\mathbf{y}\|$.
  - Define

$$\boxed{CE_k = \sum_{j=1}^{k} \frac{y_k^2}{\|\mathbf{y}\|^2}}$$

UNIVERSITY of ST. THOMAS

- Cumulative Energy is a simple vector-valued function that gives information about the concentration of energy in a vector.
- Let $\mathbf{v} = (v_1, \ldots, v_N) \in \mathbb{R}^N$.
  - Take the absolute value of each component of $\mathbf{v}$ and sort from largest to smallest. Call this new vector $\mathbf{y}$.
  - Note $\|\mathbf{v}\| = \|\mathbf{y}\|$.
  - Define

$$\boxed{CE_k = \sum_{j=1}^{k} \frac{y_k^2}{\|\mathbf{y}\|^2}}$$

University of St. Thomas

- Cumulative Energy is a simple vector-valued function that gives information about the concentration of energy in a vector.
- Let $\mathbf{v} = (v_1, \ldots, v_N) \in \mathbb{R}^N$.
  - Take the absolute value of each component of $\mathbf{v}$ and sort from largest to smallest. Call this new vector $\mathbf{y}$.
  - Note $\|\mathbf{v}\| = \|\mathbf{y}\|$.
  - Define

$$\boxed{CE_k = \sum_{j=1}^{k} \frac{y_k^2}{\|\mathbf{y}\|^2}}$$

UNIVERSITY of ST. THOMAS

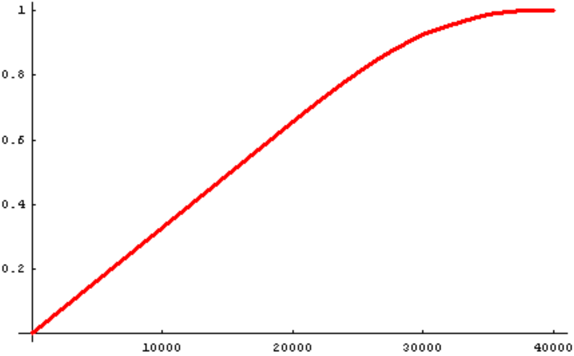
- Cumulative Energy is a simple vector-valued function that gives information about the concentration of energy in a vector.
- Let $\mathbf{v} = (v_1, \ldots, v_N) \in \mathbb{R}^N$.
  - Take the absolute value of each component of $\mathbf{v}$ and sort from largest to smallest. Call this new vector $\mathbf{y}$.
  - Note $\|\mathbf{v}\| = \|\mathbf{y}\|$.
  - Define

$$\boxed{CE_k = \sum_{j=1}^{k} \frac{y_k^2}{\|\mathbf{y}\|^2}}$$

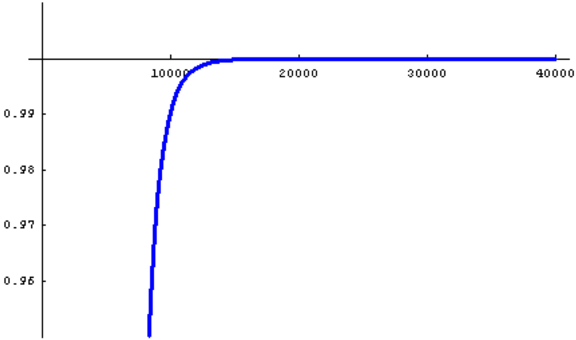
$$
\begin{aligned}
CE_1 &= \frac{y_1^2}{y_1^2 + y_2^2 + \cdots + y_N^2} \\
CE_2 &= \frac{y_1^2 + y_2^2}{y_1^2 + y_2^2 + \cdots + y_N^2} \\
\vdots \quad & \quad \vdots \\
CE_N &= \frac{y_1^2 + y_2^2 + \cdots + y_N^2}{y_1^2 + y_2^2 + \cdots + y_N^2} = 1
\end{aligned}
$$

University of St. Thomas

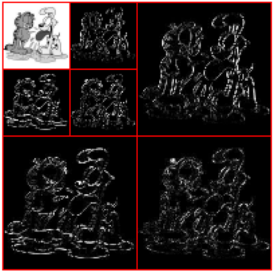
Here is the cumulative energy plot for the Garfield image:

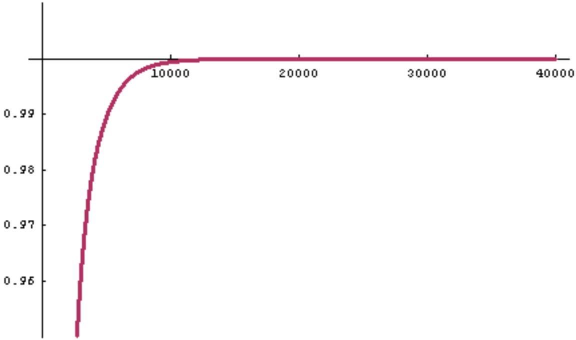


Here is the cumulative energy plot for one iteration of the transformed image:



Here is the cumulative energy plot for two iterations of the transformed image:





We will use Cumulative Energy to

- Determining an amount of energy we wish to retain.
- Identifying those elements in the vector that produce that energy.
- Converting the remaining elements to 0.

We will use Cumulative Energy to

- Determining an amount of energy we wish to retain.
- Identifying those elements in the vector that produce that energy.
- Converting the remaining elements to 0.

We will use Cumulative Energy to

- Determining an amount of energy we wish to retain.
- Identifying those elements in the vector that produce that energy.
- Converting the remaining elements to 0.

- In 1952, David Huffman made a simple observation:
- *Rather than use the same number of bits to represent each character, why not use a short bit stream for characters that appear often in an image and a longer bit stream for characters that appear infrequently in the image?*
- He then developed an algorithm to do just that. We refer to his simple algorithm as Huffman encoding. We will illustrate the algorithm via an example.

- In 1952, David Huffman made a simple observation:
- *Rather than use the same number of bits to represent each character, why not use a short bit stream for characters that appear often in an image and a longer bit stream for characters that appear infrequently in the image?*
- He then developed an algorithm to do just that. We refer to his simple algorithm as Huffman encoding. We will illustrate the algorithm via an example.

- In 1952, David Huffman made a simple observation:
- *Rather than use the same number of bits to represent each character, why not use a short bit stream for characters that appear often in an image and a longer bit stream for characters that appear infrequently in the image?*
- He then developed an algorithm to do just that. We refer to his simple algorithm as Huffman encoding. We will illustrate the algorithm via an example.

- Suppose you want to perform Huffman encoding on the word seesaws.
- First observe that s appears three times (24 bits), *e* appears twice (16 bits), and *a* and *w* each appear once (16 bits) so the total number of bits needed to represent *seesaws* is 56.

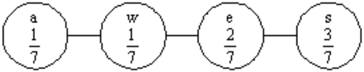
- Suppose you want to perform Huffman encoding on the word seesaws.
- First observe that s appears three times (24 bits), *e* appears twice (16 bits), and *a* and *w* each appear once (16 bits) so the total number of bits needed to represent *seesaws* is 56.

Char.	ASCII	Binary	Frequency
s	115	01110011_2	3
e	101	01100101_2	2
a	97	01100001_2	1
w	119	01110111_2	1

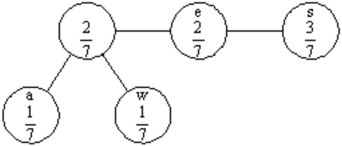
So in terms of bits, the word seesaws is

01110011 01100101 01100101 01110011 01100001 01110111 01110011

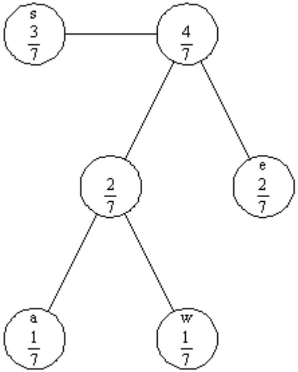
The first step in Huffman coding is as follows: Assign probabilities to each character and then sort from smallest to largest. We will put the probabilities in circles called nodes and connect them with lines (branches).



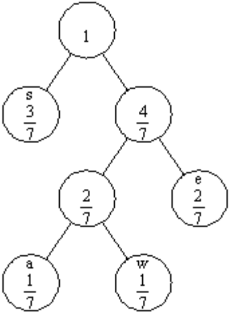
Now simply add the two smallest probabilities to create a new node with probability $2/7$. Branch the two small nodes off this one and resort the three remaining nodes:



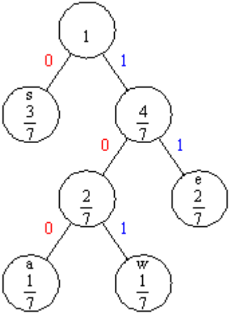
Again we add the smallest two probabilities on the top row ($2/7 + 2/7 = 4/7$), create a new node with everything below these nodes as branches and sort again:



Since only two nodes remain on top, we simply add the probabilities of these nodes together to get 1 and obtain our finished tree:


1
s
3/7
4/7
2/7
e
2/7
a
1/7
w
1/7

Now assign to each left branch the value 0 and to each right branch the value 1:


1
0
1
s
3/7
4/7
0
1
2/7
e
2/7
0
1
a
1/7
w
1/7

- We can read the new bit stream for each character right off the tree!
- Here are the new bit streams for the four characters:

- We can read the new bit stream for each character right off the tree!
- Here are the new bit streams for the four characters:

Char.	Binary
s	0_2
e	11_2
a	100_2
w	101_2

- Since *s* appears three times in *seesaws*, we need 3 bits to represent them. The character *e* appears twice (4 bits), and *a* and *w* each appear once (3 bits each).
- The total number of bits we need to represent the word *seesaws* is 13 bits! Recall without Huffman coding, we needed 56 bits so we have reduced the number of bits needed by a factor of 4!
- Here is the word *seesaws* using the Huffman codes for each character:

0111101001010

UNIVERSITY *of* ST. THOMAS

- Since *s* appears three times in *seesaws*, we need 3 bits to represent them. The character *e* appears twice (4 bits), and *a* and *w* each appear once (3 bits each).
- The total number of bits we need to represent the word *seesaws* is 13 bits! Recall without Huffman coding, we needed 56 bits so we have reduced the number of bits needed by a factor of 4!
- Here is the word *seesaws* using the Huffman codes for each character:

  0111101001010

- Since *s* appears three times in *seesaws*, we need 3 bits to represent them. The character *e* appears twice (4 bits), and *a* and *w* each appear once (3 bits each).
- The total number of bits we need to represent the word *seesaws* is 13 bits! Recall without Huffman coding, we needed 56 bits so we have reduced the number of bits needed by a factor of 4!
- Here is the word *seesaws* using the Huffman codes for each character:

$$0111101001010$$

University of St. Thomas

We are ready to perform naive image compression. Let's have a look at the notebook

`HaarImageCompression.nb`