Introduction to Lossy Image Compression: The Haar Transform

4C8: Digital Image and Video Processing

Assistant Professor François Pitié

2023/2024

Department of Electronic & Electrical Engineering , Trinity College Dublin

adapted from original material written by Prof. Anil Kokaram.

Overview

- Entropy Coding
- Haar Transform
- Quantisation
- Rate-Distortion Curves

Entropy Coding

Entropy Coding

It all starts with the entropy. The entropy of a random variable X with a probability $p(X)$ is defined by:

$$H(X) = -\sum_{x} p(x)\log_2 p(x)$$

The source coding theorem (Shannon 1948) tells that the entropy gives us the optimal average number of bits per symbol we can hope to achieve with lossless compression.

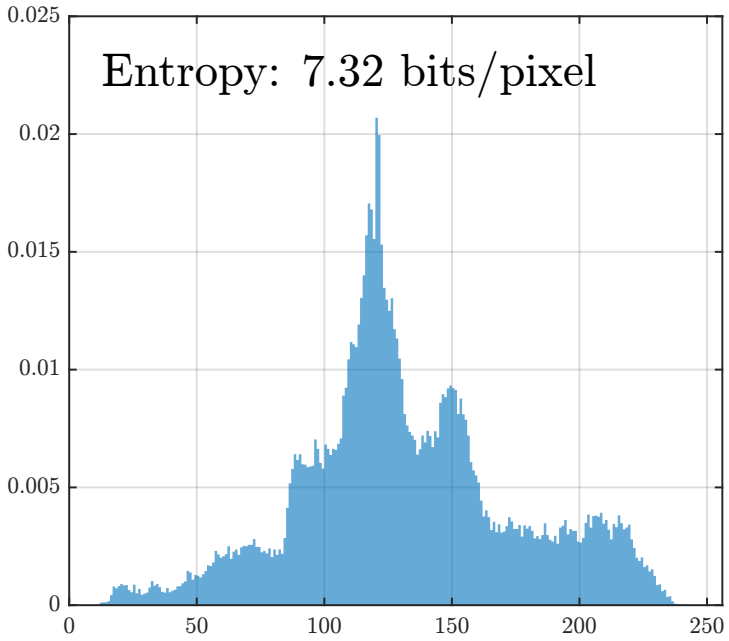
Entropy Coding

Example

Entropy operates on probabilities. Suppose that we have a horse race with eight horses taking part. Assume that the probabilities of winning for the eight horses are $(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64})$. We can calculate the entropy of the horse race as

$$
\begin{aligned}
H(X) &= -\sum_x p(x)\log_2 p(x) \\
&= -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_4\frac{1}{2} - \frac{1}{8}\log_2\frac{1}{8} - \frac{1}{16}\log_2\frac{1}{16} - 4\frac{1}{64}\log_2\frac{1}{64} \\
&= 2 \text{ bits}
\end{aligned}
$$

Entropy of an Image



The entropy of the image is 7.32 bits/pixel.

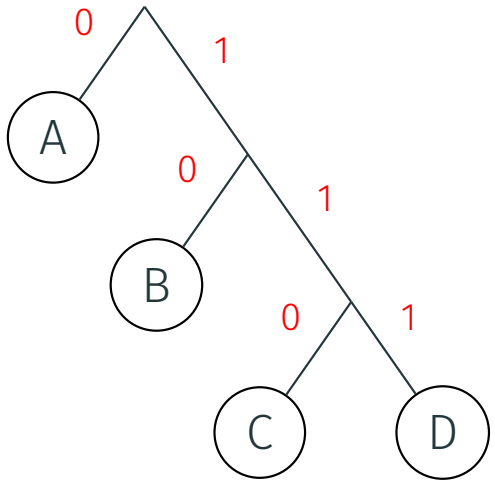
As the raw mage is encoded with 8 bits per pixel, it doesn't look like much compression should be possible here.

Huffman Coding

Huffman is the simplest entropy coding scheme. It achieves average code lengths no more than 1 bit/symbol of the entropy.

A binary tree is built by combining the two symbols with lowest probability into a dummy node.

The code length for each symbol is the number of branches between the root and respective leaf.



A:	$p(A) = 0.6$	code:	0
B:	$p(B) = 0.25$	code:	10
C:	$p(C) = 0.1$	code:	110
D:	$p(D) = 0.05$	code:	111

Huffman Coding

On our image, the table of codes is as follows:

Symbol (pixel value)	Code Length in bits
⋮	⋮
11	15
12	14
13	13
14	13
15	12
⋮	⋮

The average code length is

$$\sum_{k=0}^{255} p_k l_k = 7.35 \text{ bits/pixel}$$

The average code length is not much greater than the entropy (7.32).

Entropy Rate

Entropy is not the minimum average codeword length for a source with memory.

If the other pixel values are known we can predict the unknown pixel with much greater certainty and hence the effective (ie. conditional) entropy is much less.

The **Entropy Rate** is the minimum average codeword length for any source. It is defined as

$$H(X) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \cdots, X_n)$$

It is the limit of the joint entropy for all pixel values taken in order.

Dictionary Coding

It is very difficult to achieve codeword lengths close to the entropy rate. In fact it is difficult to calculate the entropy rate itself.

You looked at **LZW** last year as a practical coding algorithm. The average code length tends to the entropy rate if the file is large enough.

Efficiency is improved if we use Huffman to encode the output of LZW (see **Deflate**).

LZ algorithms used in lossless compression formats (eg. `.tiff`, `.png`, `.gif`, `.zip`, `.gz`, `.rar`, etc.)

Run-length encoding (RLE)

It is a form of lossless data compression, where we indicate the number of times a symbol will be repeated. The encoded stream has the following form:

```
[number of occurrences][symbol]...
```

Example

aacaacabcabaaac

becomes

2a1c2a1c1a1b1c1a1b3a1c

LZ77

It is a form of dictionary coding, where we point to previously decoded matches. The encoded stream has the following form:

`(p, q, next symbol) ...`

This tell the decoder that the next `q` symbols will be copied from previously decoded symbols at position `p` symbols back in the stream, and then insert new symbol `next symbol`.

Example

aacaacabcabaaac

becomes

(0,0,'a')(1,1,'c')(3,4,'b')(3,3,'a')(12,3,'$')

(0,0,'a') indicates that there is no previous match before `'a'`. `'$'` indicates the end of stream.

Huffman Coding

image size	256 × 256
uncompressed `tiff`	64 kB
LZW `tiff`	56 kB
Deflate (L77 + Huff) `tiff`	49 kB

image size	1920 × 720
uncompressed `tiff`	5.93 MB
LZW `tiff`	4.85 MB
Deflate (L77 + Huff) `tiff`	3.7 MB

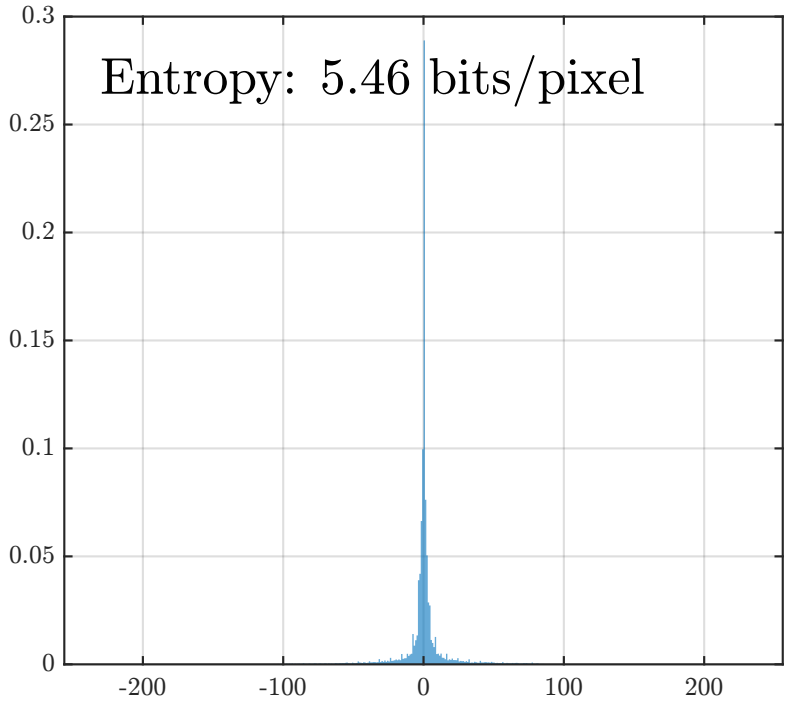
Differential Coding

$$G(x, y) = I(x, y) - I(x - 1, y)$$

The key idea is to code the spatial differences in intensity.

Differential Coding

Here is the histogram of the spatial difference image:



The entropy is now 5.46 bits/pixel, which is much less than 7.32 bits/pixel we had before (despite having twice as many symbols as the range is now -255:255 instead of 0:255).

Differential Coding

It works because the entropy of a source is:

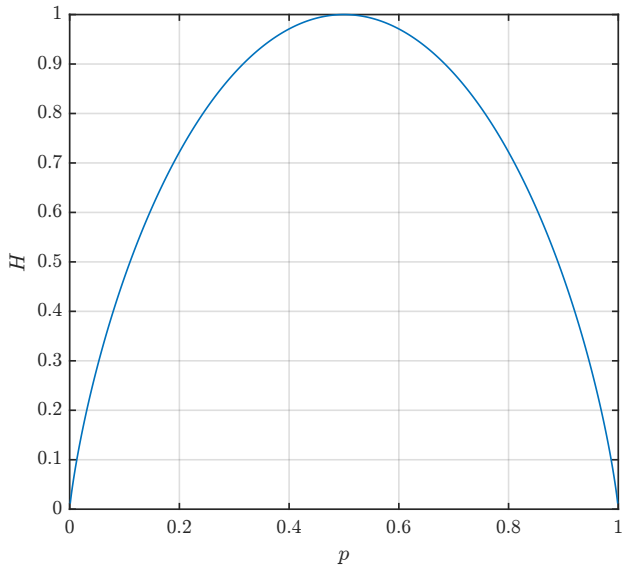
- maximised when all signals are equi-probable
- minimised when a few symbols are much more probable than the others

Differential Coding

To visualise this, consider a simple example with 2 symbols:

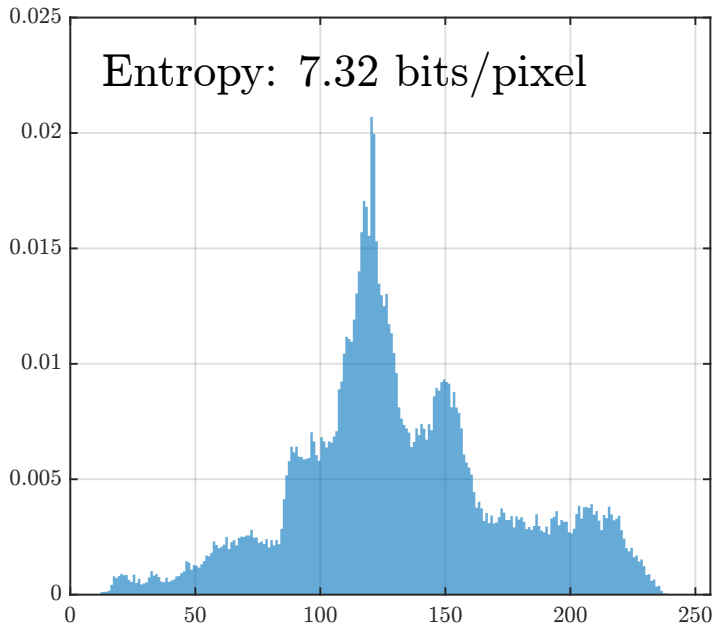
$$\begin{cases} X = 1 & \text{with probability } p \\ X = 0 & \text{with probability } 1 - p \end{cases}$$

The entropy is given by $H(X) = -p\log_2 p - (1-p)\log_2(1-p)$.

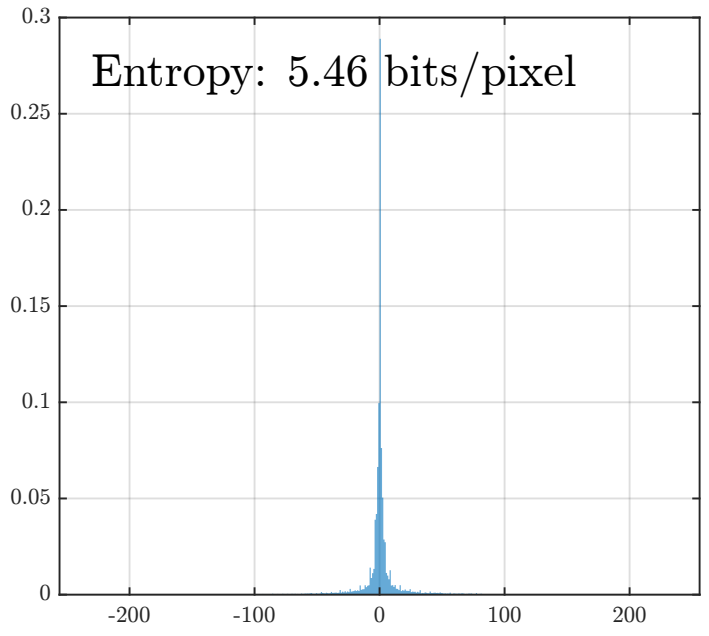


the maximum entropy is obtained at $p = 0.5$.

Differential Coding



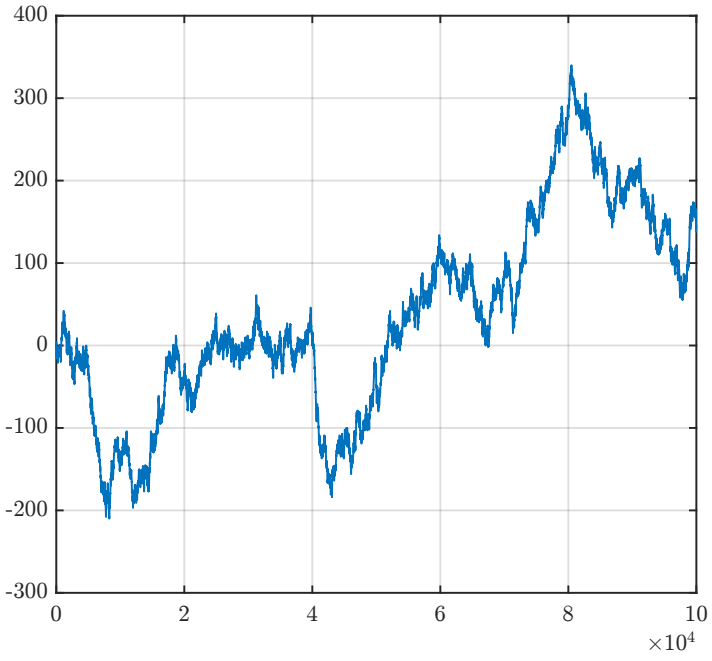
Histogram of original image.



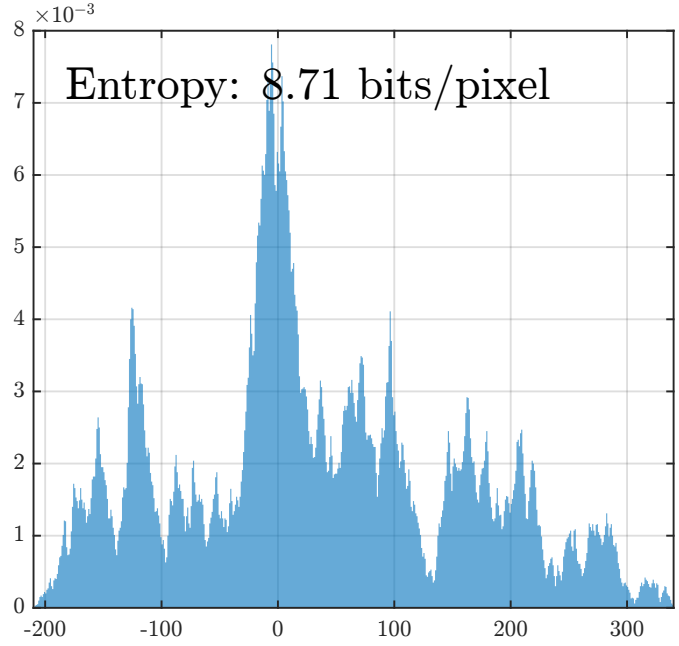
Histogram of difference image.

This simple image transform allows us to do an energy compaction (ie. compress the histogram).

Differential Coding (1D)



1D Brownian motion signal generated by random increments of ± 1.



Histogram & Entropy of that Brownian signal.

We measure an entropy of 8.71 bits/pixel but we know that the entropy rate is really 1 bit/pixel.

Lossy Compression

To go further, we need to take into account the human visual system and throw away the least perceptual details.

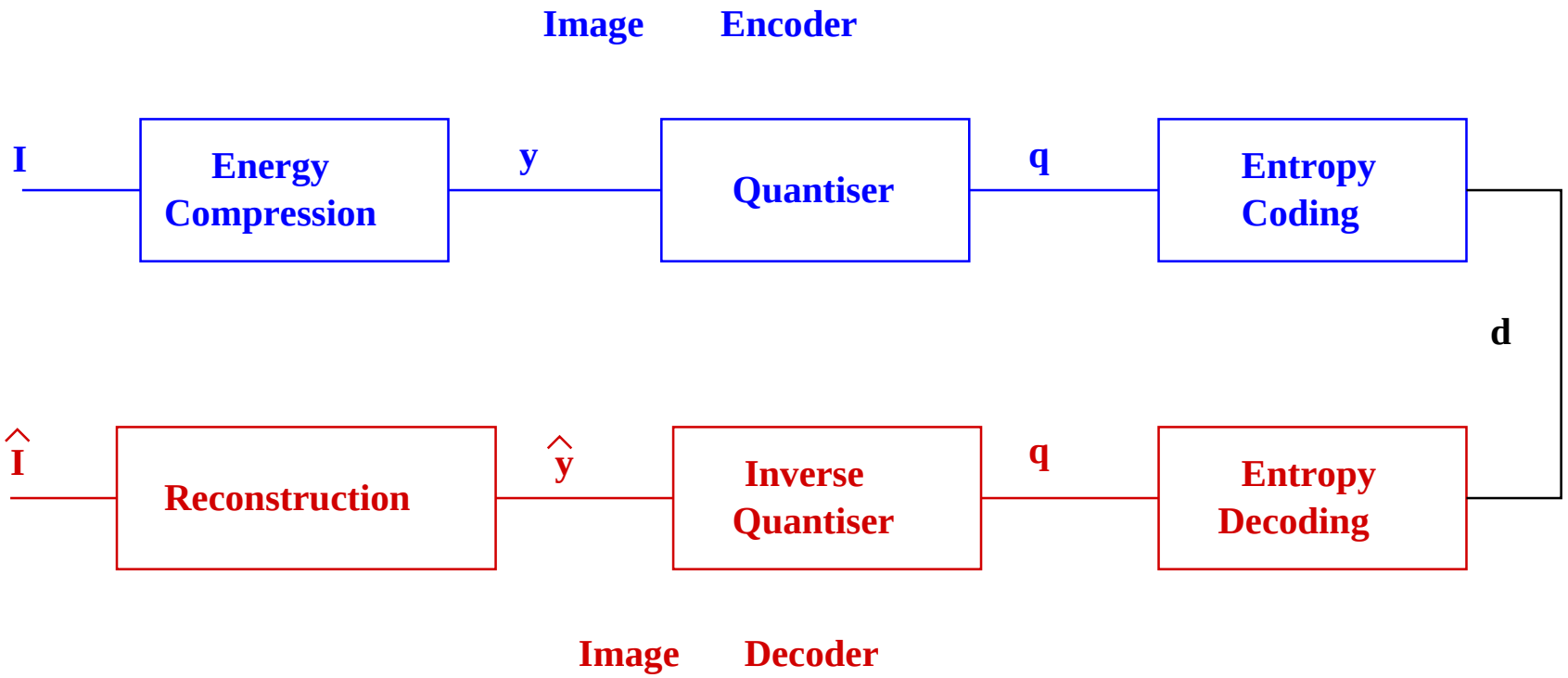
effective bit rate: 8 bits/pixel (uncompressed)

effective bit rate: 1 bits/pixel (JPEG)

Haar Transform

Lossy Compression

Here are the main blocks of a compression system:



Energy Compaction: the Haar Transform

The Haar Transform

Probably the simplest useful energy compression process is the Haar transform. In 1-dimension, this transforms a 2-element vector $[x(1), x(2)]^T$ into $[y(1), y(2)]^T$ using:

$$\begin{bmatrix} y(1) \\ y(2) \end{bmatrix} = \mathbf{T} \begin{bmatrix} x(1) \\ x(2) \end{bmatrix} \quad \text{where } \mathbf{T} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1}$$

Thus $y(1)$ and $y(2)$ are simply the sum and difference of $x(1)$ and $x(2)$, scaled by $1/\sqrt{2}$ to preserve energy.

Energy Compaction: the Haar Transform

As T is orthogonal, the inverse Haar transform is simply:

$$\begin{bmatrix} x(1) \\ x(2) \end{bmatrix} = \mathbf{T}^T \begin{bmatrix} y(1) \\ y(2) \end{bmatrix} \tag{2}$$

The Haar Transform

In 2-dimensions $\mathbf{x}$ and $\mathbf{y}$ become 2×2 matrices. We may transform first the columns of $\mathbf{x}$, by premultiplying by $\mathbf{T}$, and then the rows of the result by postmultiplying by $\mathbf{T}^T$. Hence:

$$\mathbf{y} = \mathbf{T}\,\mathbf{x}\,\mathbf{T}^T \qquad \text{and to invert:} \qquad \mathbf{x} = \mathbf{T}^T\,\mathbf{y}\,\mathbf{T}$$

The Haar Transform

To show more clearly what is happening:

$$\text{If } \mathbf{x} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \text{ then } \mathbf{y} = \frac{1}{2}\begin{bmatrix} a+b+c+d & a-b+c-d \\ a+b-c-d & a-b-c+d \end{bmatrix}$$

These operations correspond to the following filtering processes:

The Haar Transform

To show more clearly what is happening:

$$\text{If } \mathbf{x} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \text{ then } \mathbf{y} = \frac{1}{2}\begin{bmatrix} a+b+c+d & a-b+c-d \\ a+b-c-d & a-b-c+d \end{bmatrix}$$

These operations correspond to the following filtering processes:

Top left: $a+b+c+d$

It is a 4-point average or 2-D lowpass filter. It is separable: $H(z_1, z_2) = (z_1^{-1}+1)(z_2^{-1}+1)$: low pass filter along rows then low pass filter along columns.

It is called a **Lo-Lo filter**.

The Haar Transform

To show more clearly what is happening:

$$\text{If } \mathbf{x} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \text{ then } \mathbf{y} = \frac{1}{2}\begin{bmatrix} a+b+c+d & a-b+c-d \\ a+b-c-d & a-b-c+d \end{bmatrix}$$

These operations correspond to the following filtering processes:

Top right: $a - b + c - d$.

It is the average horizontal gradient. Again separable: $H(z_1, z_2) = (z_1^{-1} + 1)(z_2^{-1} - 1)$. Average along columns and difference (gradient) along rows.

Horizontal highpass and vertical lowpass: **Hi-Lo filter**.

The Haar Transform

To show more clearly what is happening:

$$\text{If } \mathbf{x} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \text{ then } \mathbf{y} = \frac{1}{2}\begin{bmatrix} a+b+c+d & a-b+c-d \\ a-c+b-d & a-b-c+d \end{bmatrix}$$

These operations correspond to the following filtering processes:

Lower left: $a + b - c - d = a - c + b - d$.

Average vertical gradient or horizontal lowpass and vertical highpass.
Again separable: $H(z_1, z_2) = (z_1^{-1} - 1)(z_2^{-1} + 1)$.

Horizontal lowpass and vertical highpass: **Lo-Hi filter**.

The Haar Transform

To show more clearly what is happening:

$$\text{If } \mathbf{x} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \text{ then } \mathbf{y} = \frac{1}{2}\begin{bmatrix} a+b+c+d & a-b+c-d \\ a-c+b-d & a-b-(c-d) \end{bmatrix}$$

These operations correspond to the following filtering processes:

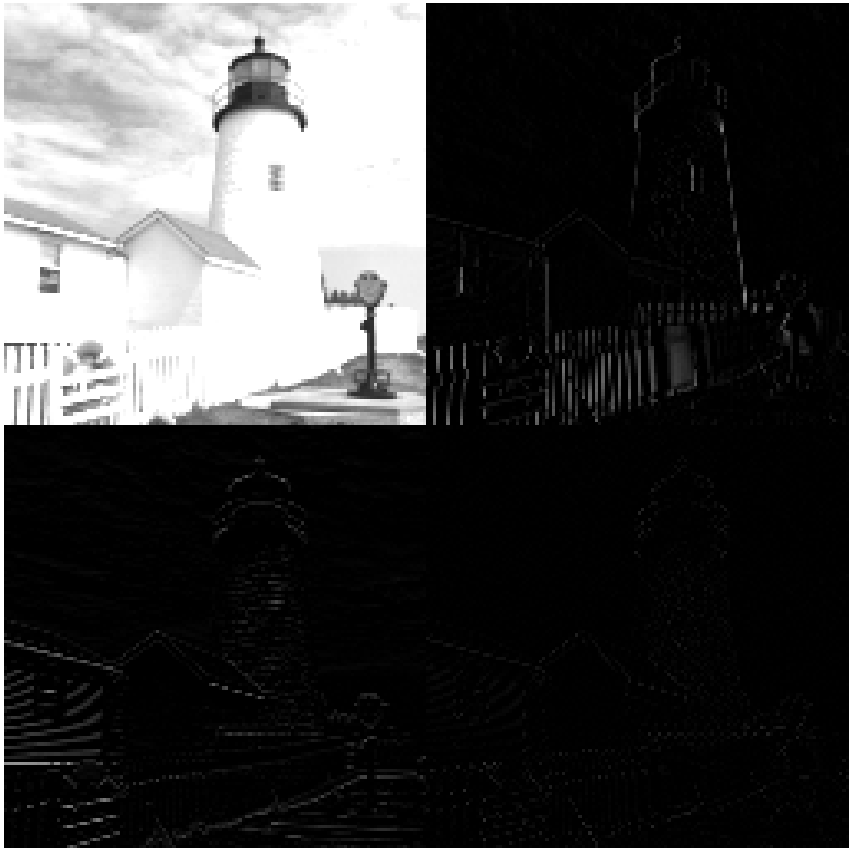
Lower right: $a - b - c + d = a - b - (c - d)$.

Again separable: $H(z_1, z_2) = (z_1^{-1} - 1)(z_2^{-1} - 1)$.

Diagonal curvature or 2-D highpass: **Hi-Hi filter**.

The Haar Transform

Original



Haar Transform

The Haar Transform

Original

Haar Transform

We can change the brightness and contrast adjusted for visualisation: LoLo /2 and 128 is added to the LoHi, HiLo and HiHi bands.

The Haar Transform

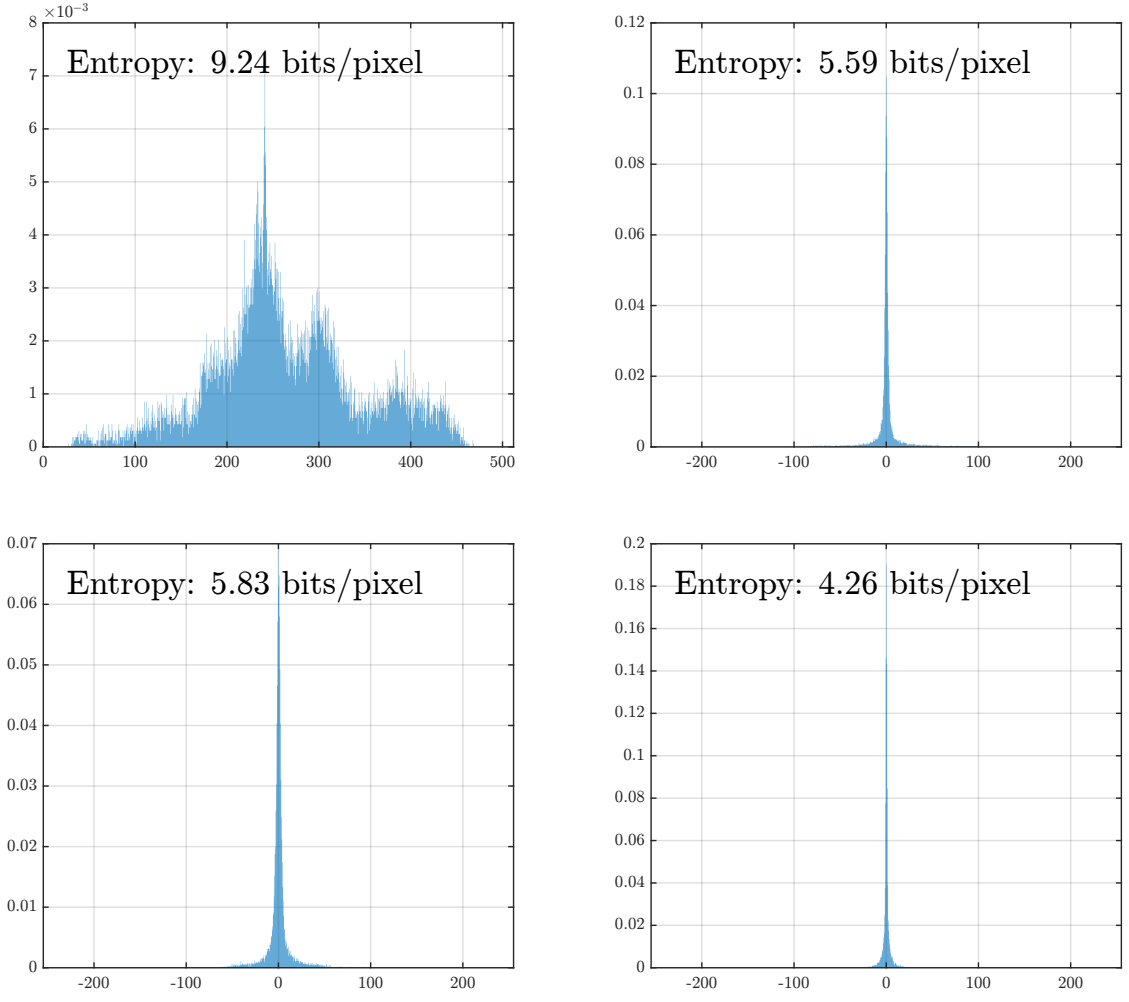


Figure: Histograms & Entropies for the 4 bands.

The Haar Transform

Calculating the overall entropy is trickier. Each coefficient in a band represents 4 pixel locations in the original image.

The entropy for the picture is thus:

$$H = \frac{H_{\text{hihi}}}{4} + \frac{H_{\text{lohi}}}{4} + \frac{H_{\text{hilo}}}{4} + \frac{H_{\text{lohi}}}{4} \approx 6.2 \text{ bits/pixel}$$

The Haar Transform

It is better, but it is not great.

The transform generates a lot of new symbols. Also the 1D Haar transform will produce irrational numbers, eg. $\frac{1}{\sqrt{2}}(a+b)$.

Quantisation

Quantisation

The missing ingredient is quantisation. It is applied to the transform coefficients:

$$Q(x) = \text{round}\left(x/Q_{step}\right)$$

The inverse quantisation is given by:

$$\hat{x} = Q_{step} \times Q(x)$$

The step size Q_{step} should normally be decided by perceptual evaluation, and we should assign different step sizes to the different bands and we can use different step sizes for the different colour channels.

For now, we will consider a uniform step size, Q_{step}, for each band.

Quantisation

Original quantised ($Q_{step} = 15$)

Haar quantised ($Q_{step} = 15$)

Quantisation

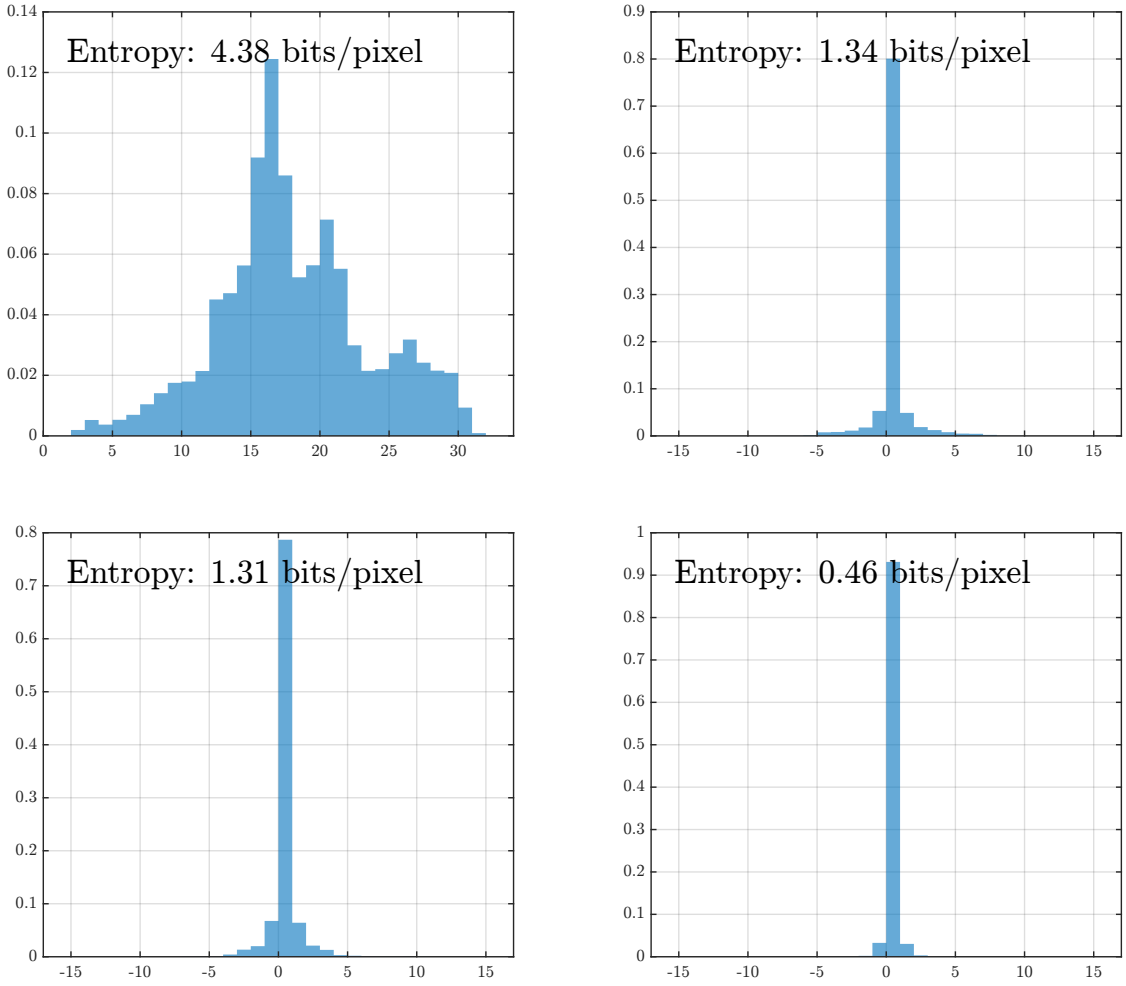


Figure: Histograms for $Q_{step} = 15$.

Quantisation

Original quantised ($Q_{step} = 15$).

Entropy: 3.45 pixels/bit

RMS error: 4.27

Haar quantised ($Q_{step} = 15$).

Entropy: 1.87 pixels/bit

RMS error: 3.26

Quantisation

Original quantised ($Q_{step} = 40$).

Entropy: 2.14 pixels/bit

RMS error: 11.12

Haar quantised ($Q_{step} = 40$).

Entropy: 1.09 pixels/bit

RMS error: 7.42

Rate-Distortion Curves

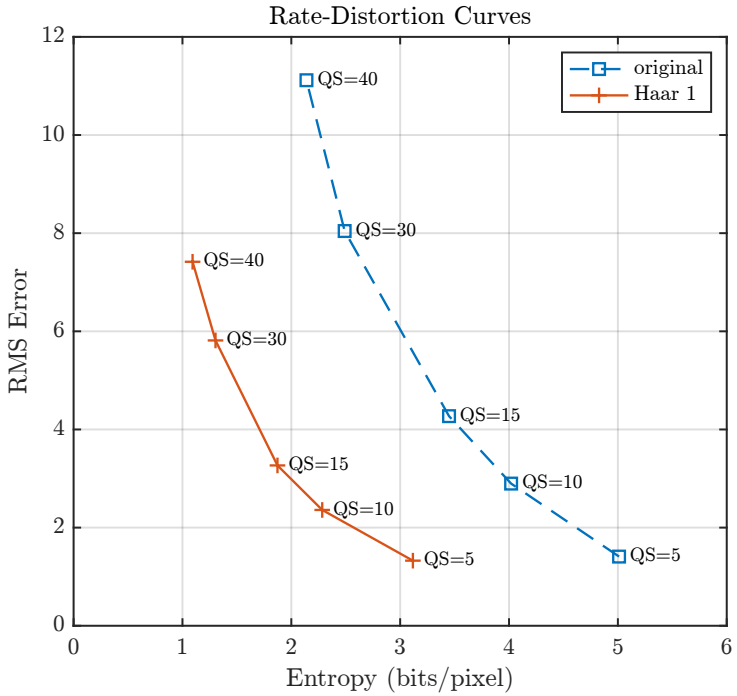
In our examples, it seems that applying quantisation lowers both the entropy and the reconstruction error.

In general, when comparing compression algorithms, we have to be mindful that we have two variables at play: the bitrate (here estimated with the entropy), and the distortion (which can be measured by any objective metric, such as PSNR, SSIM, RMS, etc.).

Lossy compression algorithms have to handle this trade-off between Distortion and Bitrate. It is easy to improve quality if we are given more bits.

The proper tool for comparing compression methods is thus to plot and compare rate-distortion curves.

Rate-Distortion Curves



Each operating point of these curves is obtained by varying one parameter of method (here the quantisation step Q_{step}).

Rate-Distortion Curves (Analysis)

The various operating points allow us to interpolate the curves (eg. for $Q_{step} \approx 20$ the RMS error for Haar is probably about 4).

One way to read this graph is that for any target RMS error, Haar can produce a compressed image at that quality level, with a lower bitrate than the corresponding quantised original image at that the same quality level.

Alternatively, we note that for any level of bitrate, Haar can produce better quality picture (eg. at 3bits/pixel, we measure a RMS of about 2.7 for Haar vs. about 6 without Haar).

Important: the quantisation steps might mean different things for both methods. What we want is to vary some parameters, so as to cover a wide range of bitrates. The Q_{step} values themselves do not matter when in an RD-curve.

Multi-Level Haar

Multi-Level Haar

Most of the entropy left is in the **LoLo** band.

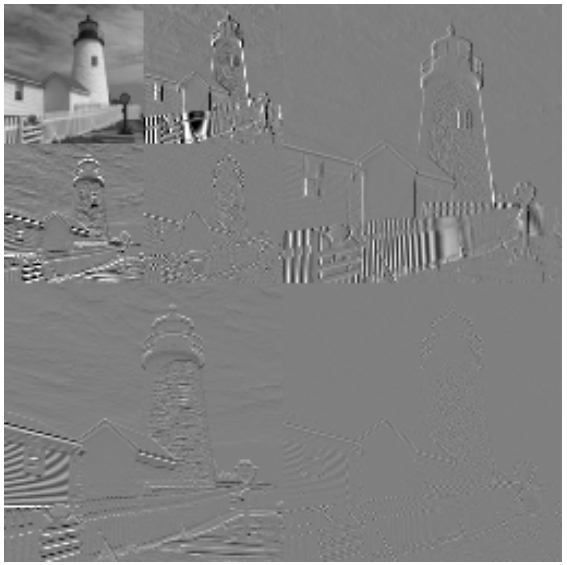
As the **LoLo** band basically a picture, we could try to apply the Haar transform to that smaller image and thus further reduce its entropy.

This is the idea behind the multi-level Haar Transform.

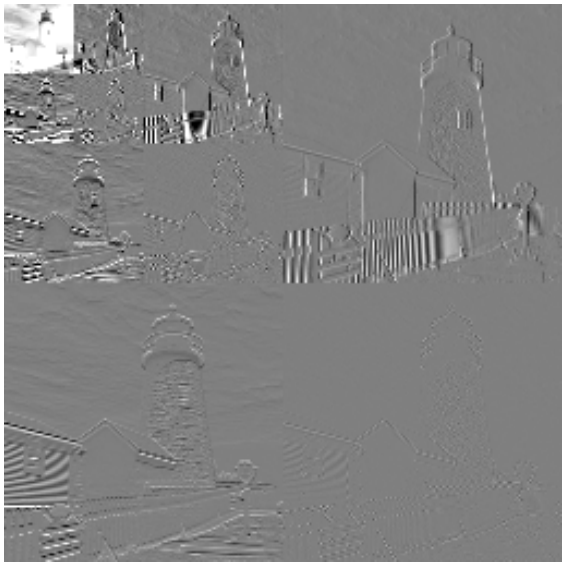
Multi-Level Haar

The multi-level Haar is obtained by iteratively applying the Haar transform to the **LoLo** band at each level.

1 level



2 levels



3 levels

note: the brightness and contrast have been changed for better visualisation.

Multi-Level Haar: Visual Comparisons

Original, $Q_{step} = 15$

Haar level 1, $Q_{step} = 15$

Multi-Level Haar: Visual Comparisons

Haar level 1, $Q_{step} = 15$

Haar level 4, $Q_{step} = 15$

Compression for levels of Haar yield smoother reconstructions.

Multi-Level Haar: Visual Comparisons

Original, $Q_{step} = 40$

Haar level 1, $Q_{step} = 40$

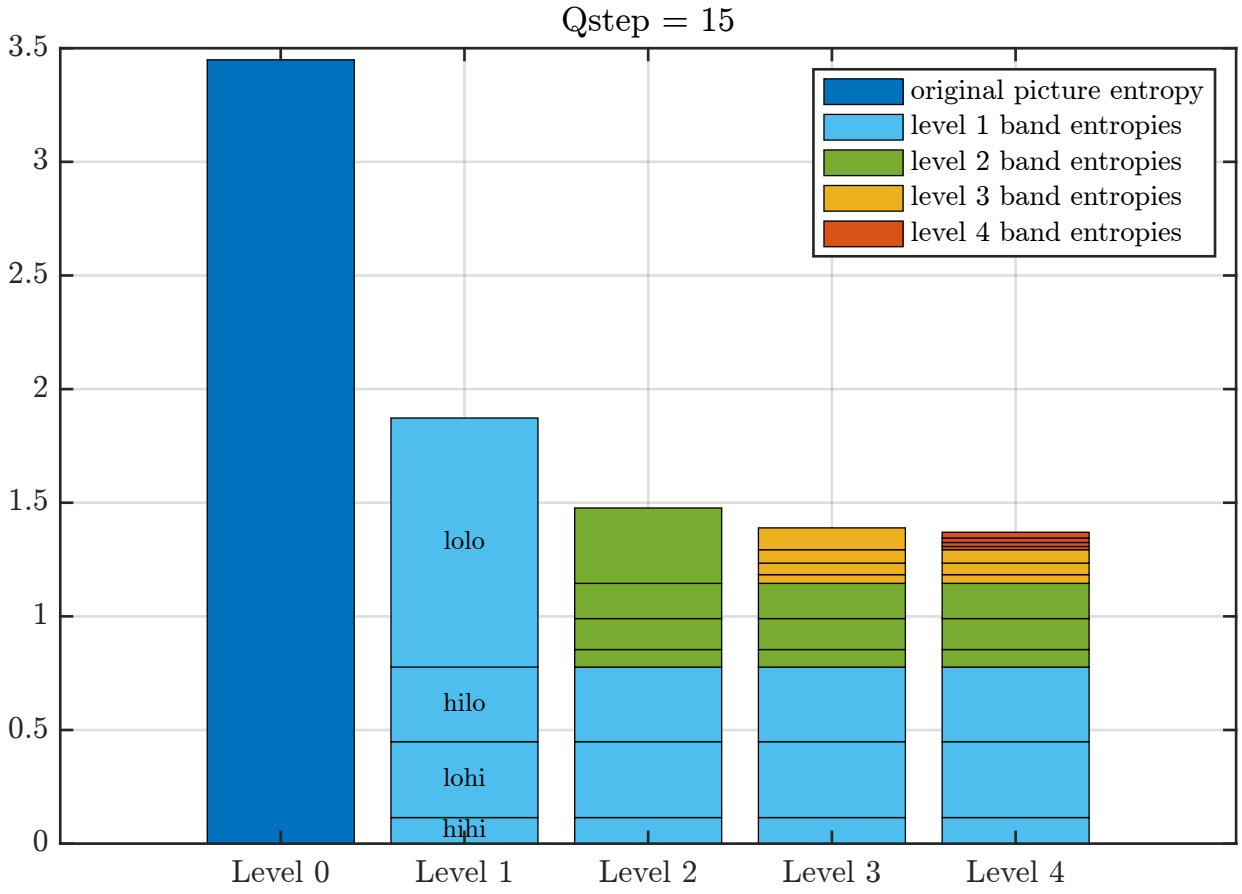
Multi-Level Haar: Visual Comparisons

Haar level 1, $Q_{step} = 40$

Haar level 4, $Q_{step} = 40$

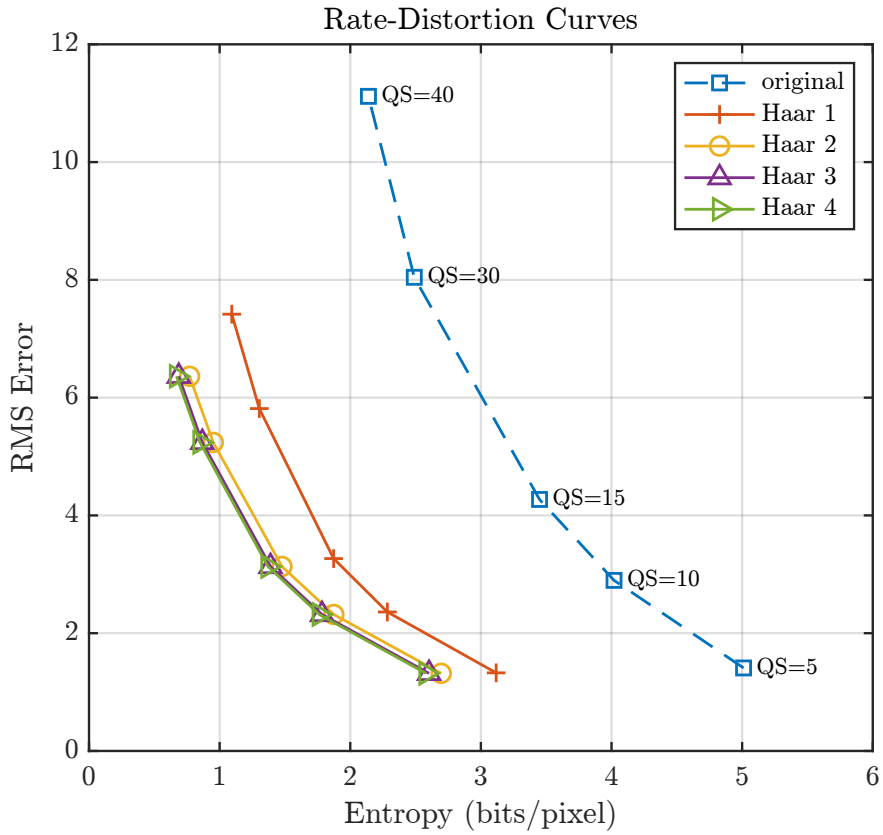
Compression for levels of Haar yield smoother reconstructions.

Multi-Level Haar: Entropy



Breakdown of the entropy contributions of the Haar bands at each level.

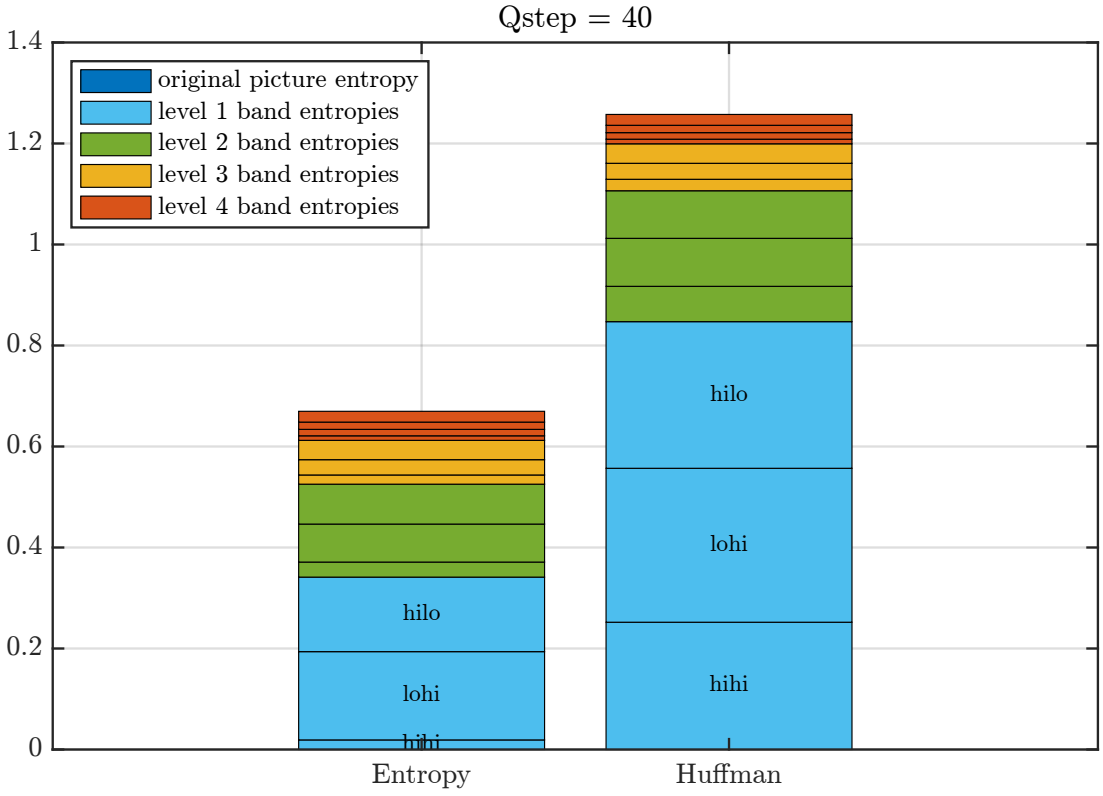
Multi-Level Haar: Rate-Distortion Curves



The RD-curve shows that we have diminishing returns when using higher levels, but that the multi-level approach is clearly beneficial.

Practical Entropy Encoding

Huffman vs. Ideal Bitrates



The ideal bit/pixel ratio given by the entropy is somewhat far from the actual performance of Huffman coding.

Why is Huffman Inefficient?

Why is the code inefficient?

Remember that for a symbol k, we have:

$$\text{length}_k = -\log_2(p_k)$$

The code is inefficient because after quantising the Haar tranform, the symbol **'0'** is very frequent, with a probability much bigger than 0.5 (0.8 approx). Hence the ideal symbol length for **'0'**:

$$\text{length}_0 = -\log_2(0.8) = 0.32\text{bits}$$

but with Huffman the minimum symbol length is 1 bit.

Run-Length Codes (RLC)

To achieve a bitrate that is lower than 1, we can use pre-process the symbols with Run-length codes (RLCs) on the 0 symbol.

That is, the pels of the subimage are scanned sequentially (usually in columns or rows) to form a long 1-dimensional vector.

Each non-zero sample is coded as a single symbol in the normal way.

Each run of consecutive zero samples (the most probable symbol) in the vector is coded as a single symbol with an associated length (as a power of 2).

Run-Length Codes (RLC): Example

Example

`-13, -5, 1, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0`

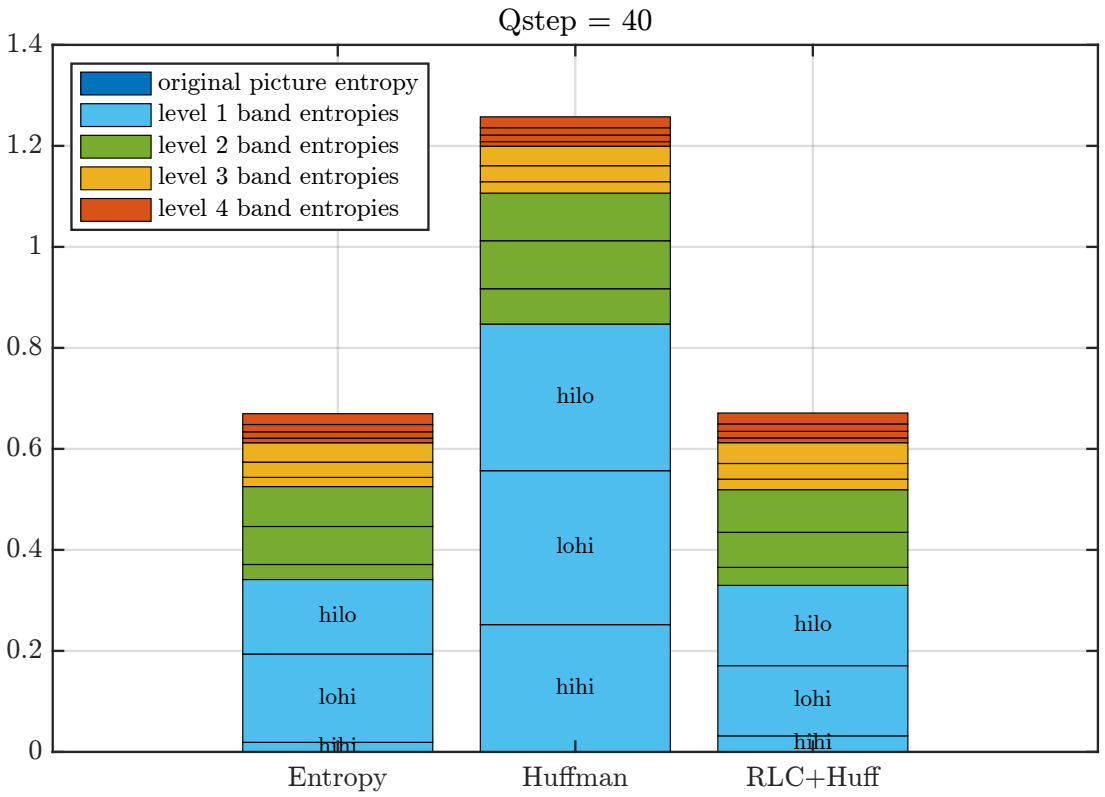
becomes

`-13, -5, 1, 4x0, 2x0, -1, 8x0, 2x0, 1x0`

`1x0`, `2x0`, `4x0`, `8x0` are the symbols for 1, 2, 4 and 8 consecutive zeros.

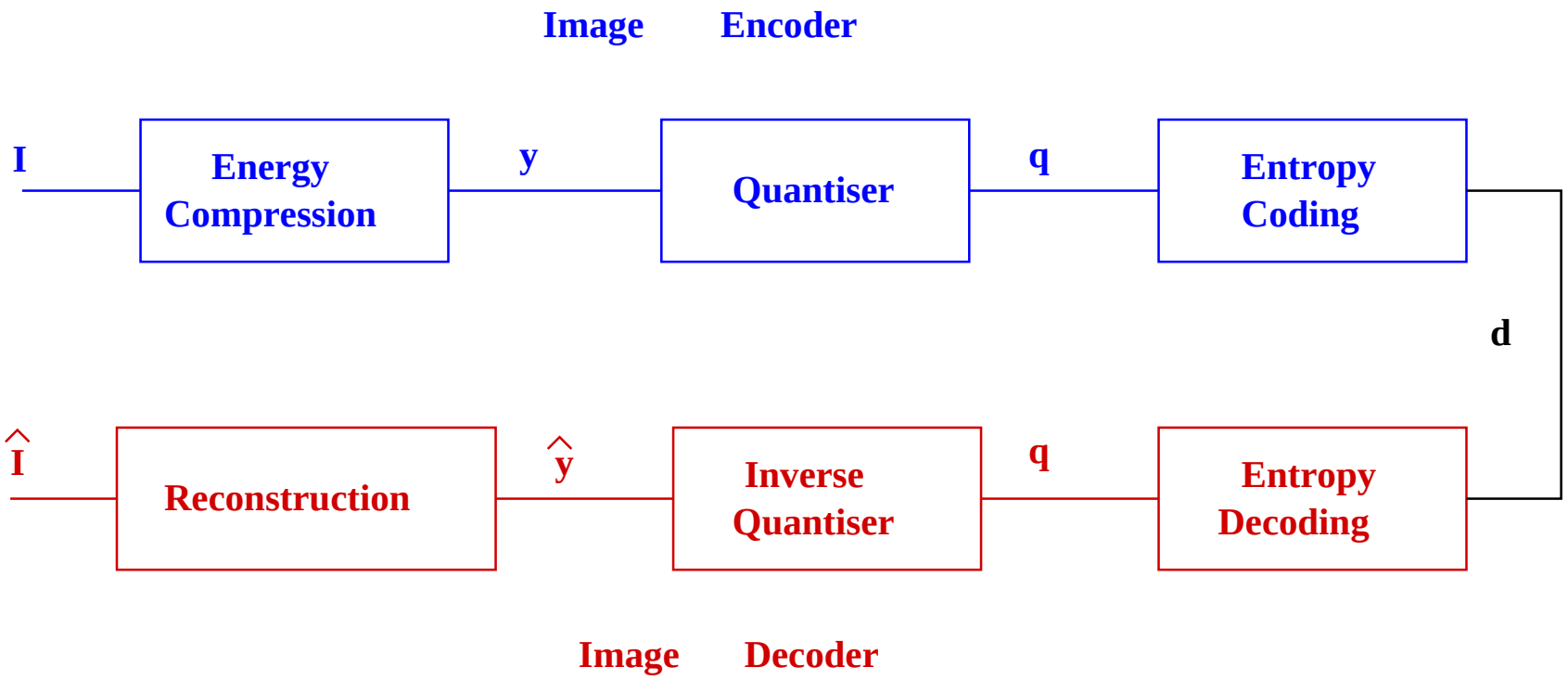
RLC+Huffman achieves near ideal entropy coding

By pre-processing the quantised transform by RLC, we are able to achieve near ideal entropy coding:



Conclusion

Conclusion



We have looked at the key components of signal compression: energy compaction through a transform, quantisation and entropy coding.

Conclusion

We have seen that the Haar Transform can be used to decrease the energy of the signal and thus achieve much better compression.

The transform leads to better picture quality. Note that we are able to achieve this **without** relying on the Human Visual System.

That is, we only looked so far at the spatial statistics of images and how they can be generated from fewer coefficients.

We are still to make better use of the Human Visual System.