Compression Algorithms for Digital Images

Adrian Delgado, Helmut Knaust

Department of Mathematical Sciences
The University of Texas at El Paso
El Paso TX 79968-0514

Tucson AZ March 31, 2012

The Problem:

We want to transform a digital image

The Problem:

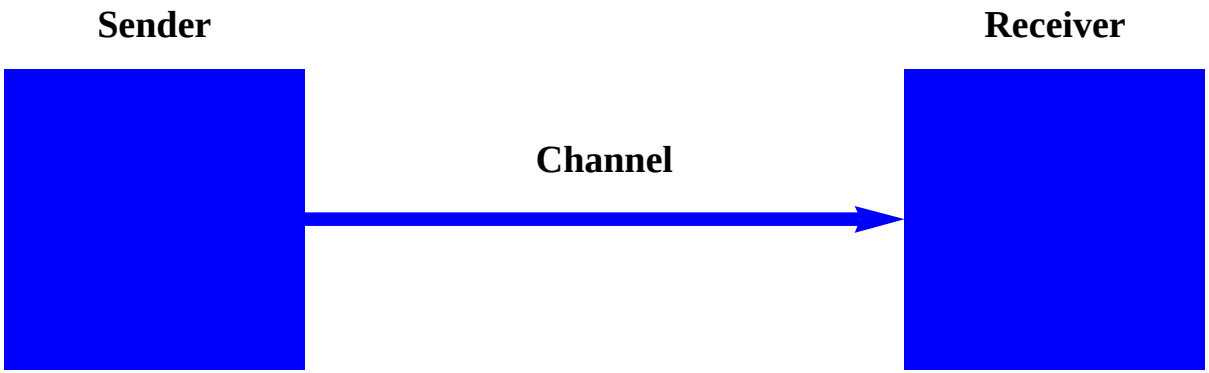
We want to transform a digital image

1. minimizing the data storage requirement for the transformed image

The Problem:

We want to transform a digital image

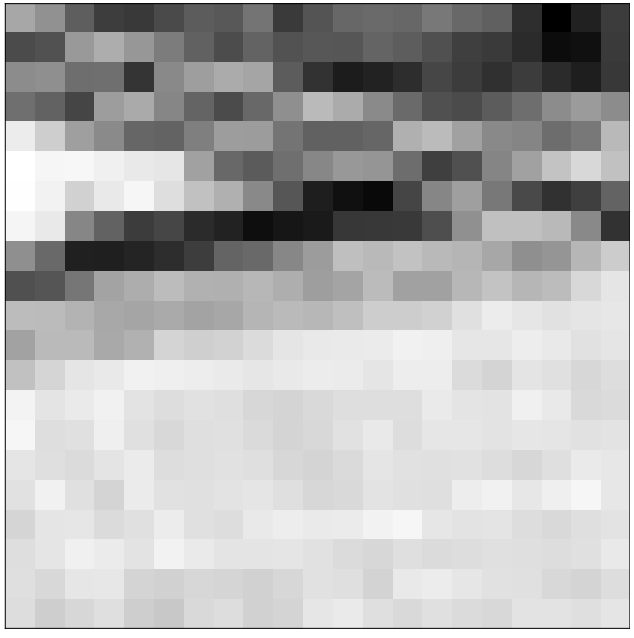
1. minimizing the data storage requirement for the transformed image
2. while maximizing the quality of the transformed image



The underlying **Information Theory** model

Each pixel in a gray-scale image is represented by an integer between 0 and $255 = 2^8 - 1$ (8 bit = 1 byte)

0=black, 255=white



$$\begin{pmatrix}
180 & 162 & 120 & 92 & 89 & 103 & 118 & 115 & 138 & 90 & 111 & 127 & 130 & 127 & 141 & 128 & 121 & 80 & 41 & 69 & 91 \\
104 & 109 & 169 & 185 & 167 & 145 & 122 & 105 & 124 & 110 & 114 & 113 & 125 & 119 & 108 & 94 & 90 & 77 & 51 & 55 & 90 \\
158 & 160 & 133 & 134 & 85 & 155 & 173 & 184 & 179 & 118 & 83 & 65 & 70 & 79 & 100 & 91 & 81 & 91 & 77 & 66 & 88 \\
134 & 123 & 99 & 172 & 183 & 153 & 125 & 104 & 128 & 160 & 196 & 184 & 156 & 130 & 108 & 104 & 118 & 133 & 158 & 170 & 158 \\
238 & 213 & 174 & 156 & 126 & 124 & 147 & 172 & 171 & 138 & 122 & 122 & 126 & 188 & 197 & 175 & 155 & 152 & 132 & 141 & 195 \\
254 & 246 & 247 & 241 & 235 & 233 & 175 & 128 & 117 & 134 & 154 & 168 & 165 & 132 & 94 & 108 & 152 & 175 & 204 & 221 & 202 \\
253 & 243 & 215 & 235 & 246 & 226 & 202 & 188 & 155 & 113 & 66 & 55 & 49 & 99 & 153 & 174 & 141 & 102 & 82 & 94 & 124 \\
245 & 235 & 153 & 123 & 91 & 100 & 77 & 70 & 53 & 60 & 63 & 87 & 88 & 89 & 106 & 161 & 201 & 201 & 195 & 155 & 83 \\
160 & 129 & 68 & 67 & 71 & 79 & 92 & 125 & 129 & 154 & 171 & 200 & 195 & 203 & 195 & 192 & 180 & 160 & 165 & 193 & 211 \\
108 & 112 & 140 & 177 & 185 & 198 & 189 & 188 & 194 & 186 & 173 & 178 & 197 & 175 & 175 & 194 & 204 & 193 & 198 & 221 & 232 \\
198 & 197 & 190 & 181 & 179 & 183 & 177 & 180 & 192 & 196 & 194 & 200 & 212 & 212 & 215 & 228 & 238 & 233 & 229 & 232 & 234 \\
176 & 196 & 196 & 182 & 189 & 218 & 214 & 216 & 224 & 232 & 235 & 236 & 237 & 242 & 240 & 233 & 233 & 239 & 235 & 229 & 232 \\
202 & 219 & 232 & 235 & 242 & 240 & 239 & 236 & 233 & 235 & 238 & 237 & 232 & 239 & 239 & 224 & 218 & 231 & 228 & 220 & 225 \\
244 & 231 & 236 & 242 & 230 & 225 & 229 & 227 & 220 & 218 & 222 & 226 & 226 & 226 & 236 & 231 & 230 & 241 & 235 & 222 & 224 \\
246 & 226 & 228 & 240 & 228 & 221 & 227 & 228 & 223 & 218 & 221 & 229 & 235 & 225 & 233 & 233 & 230 & 233 & 232 & 229 & 230 \\
232 & 227 & 224 & 231 & 237 & 225 & 227 & 229 & 227 & 220 & 218 & 223 & 232 & 229 & 228 & 229 & 225 & 220 & 227 & 236 & 234 \\
229 & 242 & 228 & 218 & 237 & 229 & 228 & 230 & 232 & 227 & 220 & 222 & 230 & 228 & 227 & 239 & 242 & 234 & 240 & 246 & 234 \\
219 & 232 & 233 & 224 & 228 & 238 & 228 & 225 & 235 & 239 & 236 & 237 & 243 & 246 & 233 & 230 & 231 & 225 & 222 & 227 & 230 \\
226 & 232 & 241 & 237 & 230 & 243 & 236 & 231 & 231 & 232 & 229 & 224 & 220 & 227 & 224 & 225 & 228 & 227 & 225 & 228 & 235 \\
227 & 221 & 233 & 234 & 218 & 215 & 220 & 219 & 215 & 220 & 229 & 227 & 217 & 235 & 238 & 233 & 229 & 228 & 221 & 218 & 225 \\
225 & 213 & 220 & 227 & 213 & 208 & 222 & 225 & 215 & 218 & 233 & 236 & 227 & 222 & 228 & 224 & 221 & 230 & 230 & 227 & 232
\end{pmatrix}$$

- We will measure the information content of a signal/image by its *entropy* as defined by *Claude Shannon*:

- We will measure the information content of a signal/image by its *entropy* as defined by *Claude Shannon*:
- Given a signal $\vec{v} = \{v_1, v_2, \ldots, v_n\}$, let $p(v_k)$ denote the relative frequency of v_k. Then

$$Ent(\vec{v}) = \sum -p(v_k)\log_2(p(v_k))$$

- We will measure the information content of a signal/image by its *entropy* as defined by *Claude Shannon*:
- Given a signal $\vec{v} = \{v_1, v_2, \ldots, v_n\}$, let $p(v_k)$ denote the relative frequency of v_k. Then

$$Ent(\vec{v}) = \sum -p(v_k)\log_2(p(v_k))$$

- Example: For $\vec{v} = \{1, 1, 2, 2, 3\}$,

$$p(1) = p(2) = 2/5,\ p(3) = 1/5,$$

so $Ent(\vec{v}) = 2 \cdot (-2/5\log_2(2/5)) - 1/5\log_2(1/5) \approx 1.522.$

- The entropy of a signal, measured in bits per data point (pixel), is a theoretical lower bound for the data storage requirement of a signal **in its current form**.

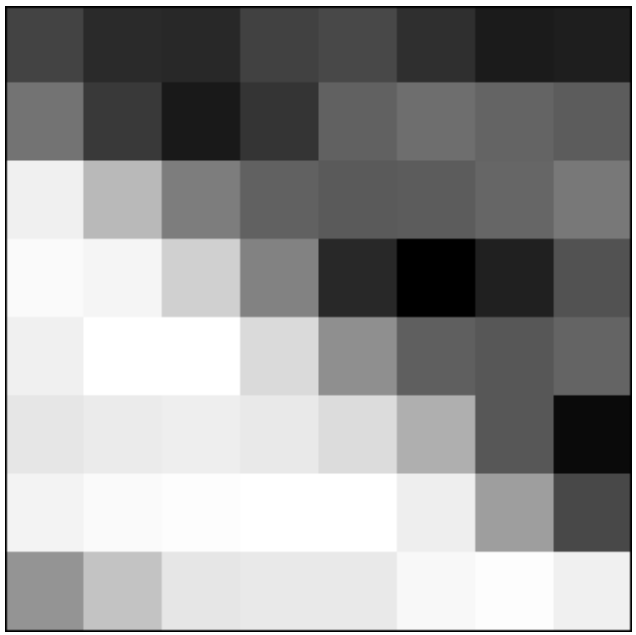
- The entropy of a signal, measured in bits per data point (pixel), is a theoretical lower bound for the data storage requirement of a signal **in its current form**.
- For the example on the previous slide $\vec{v} = \{1, 1, 2, 2, 3\}$, $Ent(\vec{v}) \approx 1.522$ bits per pixel. Thus the storage requirement is at least $5 \cdot 1.522 = 7.610$ bits.

- The entropy of a signal, measured in bits per data point (pixel), is a theoretical lower bound for the data storage requirement of a signal **in its current form**.
- For the example on the previous slide $\vec{v} = \{1, 1, 2, 2, 3\}$, $Ent(\vec{v}) \approx 1.522$ bits per pixel. Thus the storage requirement is at least $5 \cdot 1.522 = 7.610$ bits.
- There are encoding algorithms, such as **Huffman** encoding, that come pretty close to the lower bound given by the entropy.

For the **JPEG algorithm**, the image is split into sub-images of size 8×8 pixels and 128 is subtracted from each pixel.

Original image, entropy: 7.67

Here is a typical 8×8 block B:



$$\begin{pmatrix} 52 & 42 & 41 & 51 & 54 & 44 & 36 & 37 \\ 71 & 48 & 35 & 46 & 64 & 69 & 65 & 62 \\ 121 & 99 & 75 & 64 & 61 & 62 & 66 & 73 \\ 125 & 123 & 108 & 77 & 41 & 25 & 38 & 58 \\ 121 & 127 & 127 & 112 & 82 & 63 & 60 & 65 \\ 117 & 119 & 120 & 118 & 113 & 95 & 60 & 29 \\ 122 & 125 & 126 & 127 & 127 & 120 & 88 & 54 \\ 84 & 103 & 117 & 118 & 118 & 124 & 126 & 121 \end{pmatrix}$$

Below is the fixed orthogonal matrix U used by JPEG for image transformation. The matrix is based on the discrete cosine transform (DCT).

$$\begin{pmatrix}
\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\
\frac{1}{2}\cos\left(\frac{\pi}{16}\right) & \frac{1}{2}\cos\left(\frac{3\pi}{16}\right) & \frac{1}{2}\sin\left(\frac{3\pi}{16}\right) & \frac{1}{2}\sin\left(\frac{\pi}{16}\right) & -\frac{1}{2}\sin\left(\frac{\pi}{16}\right) & -\frac{1}{2}\sin\left(\frac{3\pi}{16}\right) & -\frac{1}{2}\cos\left(\frac{3\pi}{16}\right) & -\frac{1}{2}\cos\left(\frac{\pi}{16}\right) \\
\frac{1}{2}\cos\left(\frac{\pi}{8}\right) & \frac{1}{2}\sin\left(\frac{\pi}{8}\right) & -\frac{1}{2}\sin\left(\frac{\pi}{8}\right) & -\frac{1}{2}\cos\left(\frac{\pi}{8}\right) & -\frac{1}{2}\cos\left(\frac{\pi}{8}\right) & -\frac{1}{2}\sin\left(\frac{\pi}{8}\right) & \frac{1}{2}\sin\left(\frac{\pi}{8}\right) & \frac{1}{2}\cos\left(\frac{\pi}{8}\right) \\
\frac{1}{2}\cos\left(\frac{3\pi}{16}\right) & -\frac{1}{2}\sin\left(\frac{\pi}{16}\right) & -\frac{1}{2}\cos\left(\frac{\pi}{16}\right) & -\frac{1}{2}\sin\left(\frac{3\pi}{16}\right) & \frac{1}{2}\sin\left(\frac{3\pi}{16}\right) & \frac{1}{2}\cos\left(\frac{\pi}{16}\right) & \frac{1}{2}\sin\left(\frac{\pi}{16}\right) & -\frac{1}{2}\cos\left(\frac{3\pi}{16}\right) \\
\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\
\frac{1}{2}\sin\left(\frac{3\pi}{16}\right) & -\frac{1}{2}\cos\left(\frac{\pi}{16}\right) & \frac{1}{2}\sin\left(\frac{\pi}{16}\right) & \frac{1}{2}\cos\left(\frac{3\pi}{16}\right) & -\frac{1}{2}\cos\left(\frac{3\pi}{16}\right) & -\frac{1}{2}\sin\left(\frac{\pi}{16}\right) & \frac{1}{2}\cos\left(\frac{\pi}{16}\right) & -\frac{1}{2}\sin\left(\frac{3\pi}{16}\right) \\
\frac{1}{2}\sin\left(\frac{\pi}{8}\right) & -\frac{1}{2}\cos\left(\frac{\pi}{8}\right) & \frac{1}{2}\cos\left(\frac{\pi}{8}\right) & -\frac{1}{2}\sin\left(\frac{\pi}{8}\right) & -\frac{1}{2}\sin\left(\frac{\pi}{8}\right) & \frac{1}{2}\cos\left(\frac{\pi}{8}\right) & -\frac{1}{2}\cos\left(\frac{\pi}{8}\right) & \frac{1}{2}\sin\left(\frac{\pi}{8}\right) \\
\frac{1}{2}\sin\left(\frac{\pi}{16}\right) & -\frac{1}{2}\sin\left(\frac{3\pi}{16}\right) & \frac{1}{2}\cos\left(\frac{3\pi}{16}\right) & -\frac{1}{2}\cos\left(\frac{\pi}{16}\right) & \frac{1}{2}\cos\left(\frac{\pi}{16}\right) & -\frac{1}{2}\cos\left(\frac{3\pi}{16}\right) & \frac{1}{2}\sin\left(\frac{3\pi}{16}\right) & -\frac{1}{2}\sin\left(\frac{\pi}{16}\right)
\end{pmatrix}$$

We form the matrix UBU^T:

$$\begin{pmatrix}
670.125 & 107.011 & -11.7214 & 6.94264 & 1.125 & 0.873499 & 0.119709 & -1.01171 \\
-179.428 & -15.4957 & 53.6356 & 7.96463 & 21.9641 & -1.08808 & 0.181807 & 0.583231 \\
-16.6278 & -103.815 & -30.1144 & 28.4137 & -1.50951 & -0.328102 & 0.183058 & 0.547352 \\
-24.717 & 33.6658 & -53.9628 & 12.8416 & 0.634492 & 0.436325 & 0.386054 & -0.0269615 \\
-15.125 & -1.04743 & 15.7996 & -46.4442 & 1.875 & 0.122299 & -0.726564 & -0.111178 \\
-9.01396 & 46.647 & -0.0111492 & -1.86177 & -1.88268 & -0.169144 & 0.0369977 & -0.304798 \\
1.34022 & 0.151331 & -1.06694 & -0.182008 & -0.051235 & -1.04071 & 0.114407 & 0.46643 \\
29.0667 & -1.71487 & 0.796766 & 1.05864 & 0.971544 & 0.844572 & 0.207934 & -1.17675
\end{pmatrix}$$

Rounded to integers, its entropy is 3.76.

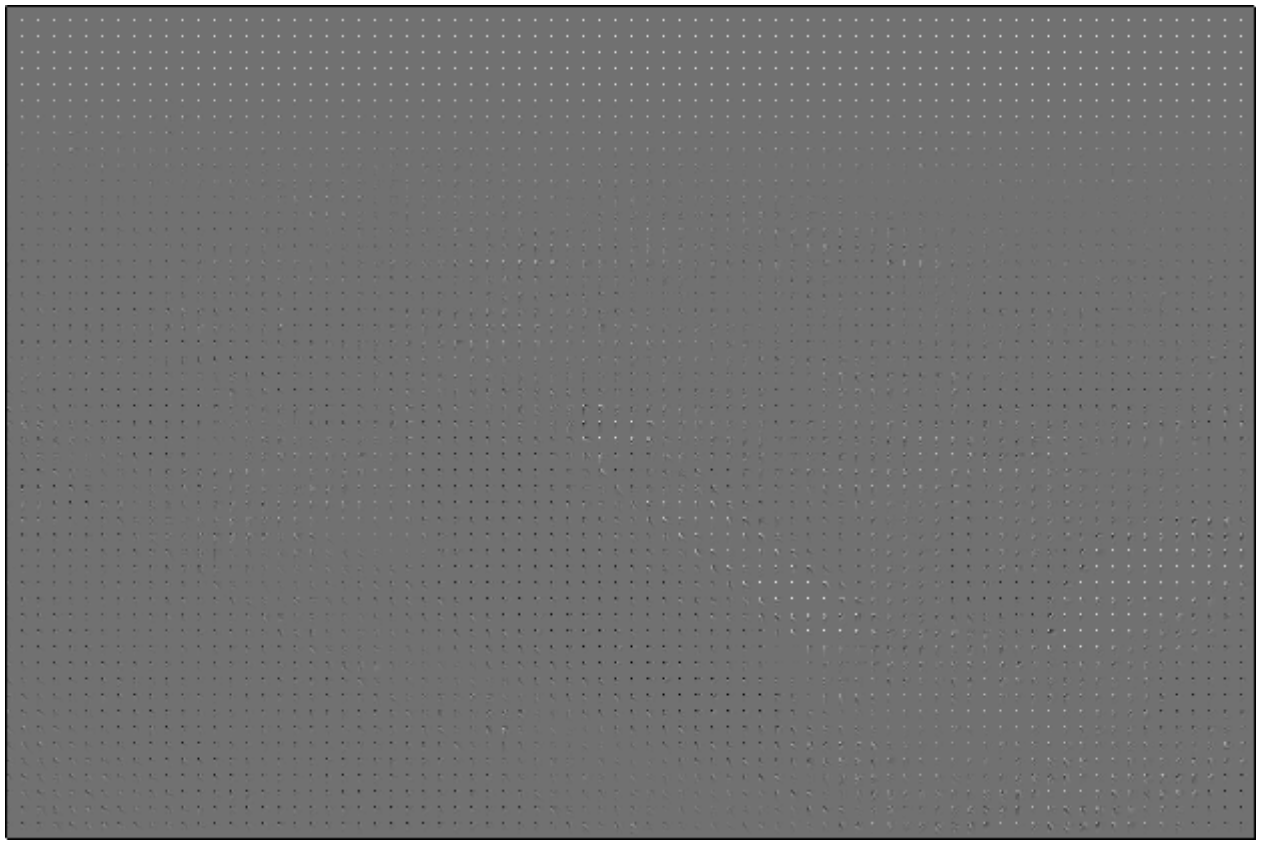
Next is a quantization step. **This step is not reversible.** We divide each entry of the transformed matrix UBU^T by the corresponding matrix entry of a normalization matrix Z, then round to the next integer. One example of Z:

$$\begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

For UBU^T, quantization of the block yields the matrix B'. (Note that the gray values are scaled: 42 is now white, -15 is black.)

$$\begin{pmatrix} 42 & 10 & -1 & 0 & 0 & 0 & 0 & 0 \\ -15 & -1 & 4 & 0 & 1 & 0 & 0 & 0 \\ -1 & -8 & -2 & 1 & 0 & 0 & 0 & 0 \\ -2 & 2 & -2 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Doing this for each 8×8 block yields the transformed quantized image. Its entropy is 1.40, leading to a compression factor of about 5.5.



- This matrix is encoded and sent to the receiver.

- This matrix is encoded and sent to the receiver.
- The receiver multiplies each block pointwise by the transformation matrix Z to obtain a matrix B''.

- This matrix is encoded and sent to the receiver.
- The receiver multiplies each block pointwise by the transformation matrix Z to obtain a matrix B''.
- The transformation U is applied to obtain $U^T B'' U$.

- This matrix is encoded and sent to the receiver.
- The receiver multiplies each block pointwise by the transformation matrix Z to obtain a matrix B''.
- The transformation U is applied to obtain $U^T B'' U$.
- Finally 128 is added to each pixel.

Here is the recovered image:

- To compare the original image to the recovered image, we use a logarithmic scale based on the average square error. For two matrices A and B of dimensions $M \times N$ we define the **Peak Signal to Noise Ratio**

$$PSNR(A, B) = 10 \log_{10} \left(\frac{255^2}{Err(A, B)} \right),$$

where

$$Err(A, B) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (A_{ij} - B_{ij})^2.$$

- To compare the original image to the recovered image, we use a logarithmic scale based on the average square error. For two matrices A and B of dimensions $M \times N$ we define the **Peak Signal to Noise Ratio**

$$PSNR(A, B) = 10 \log_{10} \left(\frac{255^2}{Err(A, B)} \right),$$

where

$$Err(A, B) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (A_{ij} - B_{ij})^2.$$

- In our example the PSNR between the original and the recovered image is 29.4. (If $A = B$, their PSNR is ∞.)

- The **JPEG2000 algorithm** applies a biorthogonal wavelet matrix pair to the whole image matrix.

- The **JPEG2000 algorithm** applies a biorthogonal wavelet matrix pair to the whole image matrix.
- The matrices W and $\tilde{W}$ used satisfy the orthogonality condition

$$\tilde{W}^T W = Id.$$

- The **JPEG2000 algorithm** applies a biorthogonal wavelet matrix pair to the whole image matrix.
- The matrices W and $\tilde{W}$ used satisfy the orthogonality condition
$$\tilde{W}^T W = Id.$$
- The matrices use filters of length 7 and 9, hence their name: *CDF97*.

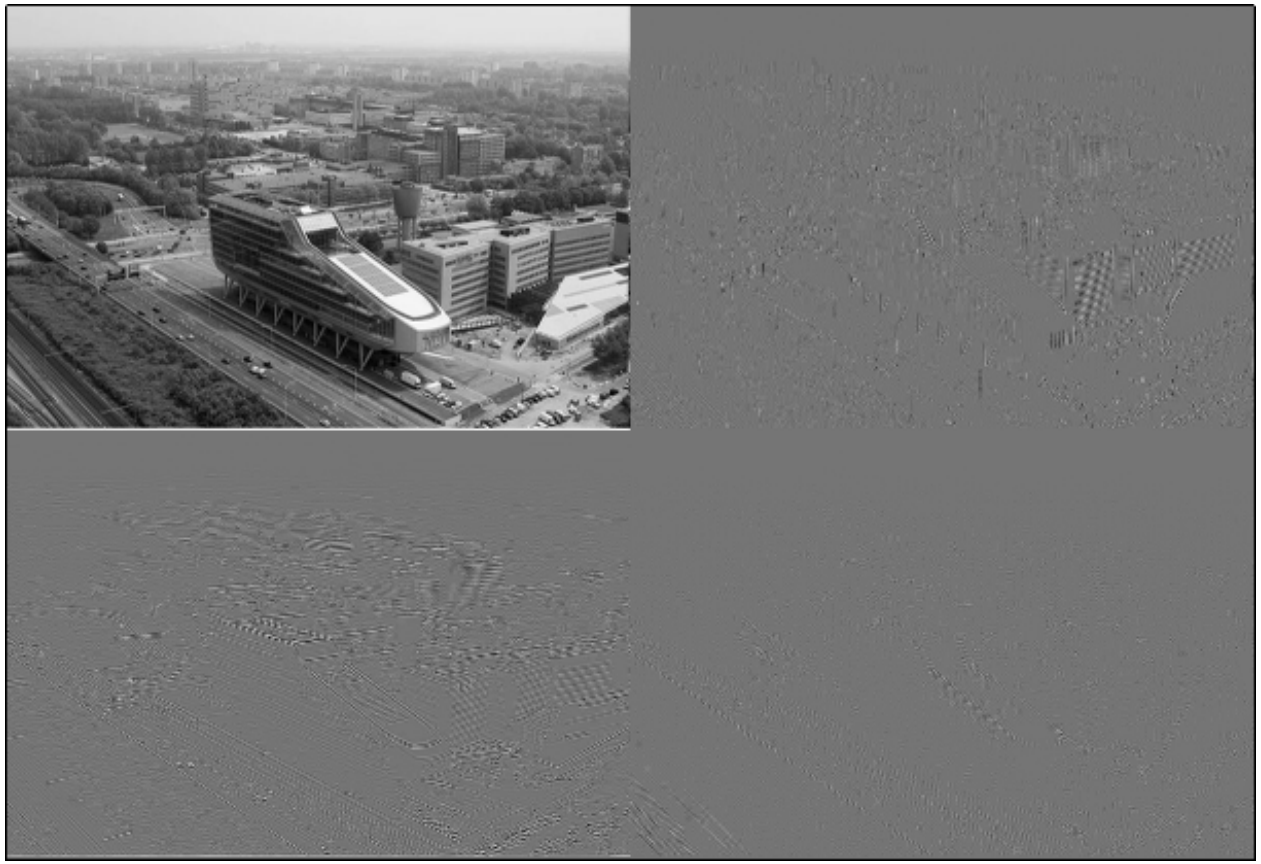
The matrix W has the general shape

$$\begin{pmatrix}
-0.0645389 & -0.0406894 & 0.418092 & 0.788486 & 0.418092 & -0.0406894 & -0.0645389 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -0.0645389 & -0.0406894 & 0.418092 & 0.788486 & 0.418092 & -0.0406894 & -0.0645389 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -0.0645389 & -0.0406894 & 0.418092 & 0.788486 & 0.418092 & -0.0406894 & -0.0645389 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -0.0645389 & -0.0406894 & 0.418092 & 0.788486 & 0.418092 & -0.0406894 & -0.0645389 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.0645389 & -0.0406894 & 0.418092 & 0.788486 & 0.418092 & -0.0406894 & -0.0645389 & 0 \\
-0.0645389 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.0645389 & -0.0406894 & 0.418092 & 0.788486 & 0.418092 & -0.0406894 \\
0.418092 & -0.0406894 & -0.0645389 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.0645389 & -0.0406894 & 0.418092 & 0.788486 \\
0.418092 & 0.788486 & 0.418092 & -0.0406894 & -0.0645389 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.0645389 & -0.0406894 \\
0.0378285 & 0.0238495 & -0.110624 & -0.377403 & 0.852699 & -0.377403 & -0.110624 & 0.0238495 & 0.0378285 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.0378285 & 0.0238495 & -0.110624 & -0.377403 & 0.852699 & -0.377403 & -0.110624 & 0.0238495 & 0.0378285 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.0378285 & 0.0238495 & -0.110624 & -0.377403 & 0.852699 & -0.377403 & -0.110624 & 0.0238495 & 0.0378285 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.0378285 & 0.0238495 & -0.110624 & -0.377403 & 0.852699 & -0.377403 & -0.110624 & 0.0238495 & 0.0378285 & 0 \\
0.0378285 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0378285 & 0.0238495 & -0.110624 & -0.377403 & 0.852699 & -0.377403 & -0.110624 & 0.0238495 \\
-0.110624 & 0.0238495 & 0.0378285 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0378285 & 0.0238495 & -0.110624 & -0.377403 & 0.852699 & -0.377403 \\
0.852699 & -0.377403 & -0.110624 & 0.0238495 & 0.0378285 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0378285 & 0.0238495 & -0.110624 & -0.377403 \\
-0.110624 & -0.377403 & 0.852699 & -0.377403 & -0.110624 & 0.0238495 & 0.0378285 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0378285 & 0.0238495
\end{pmatrix}$$

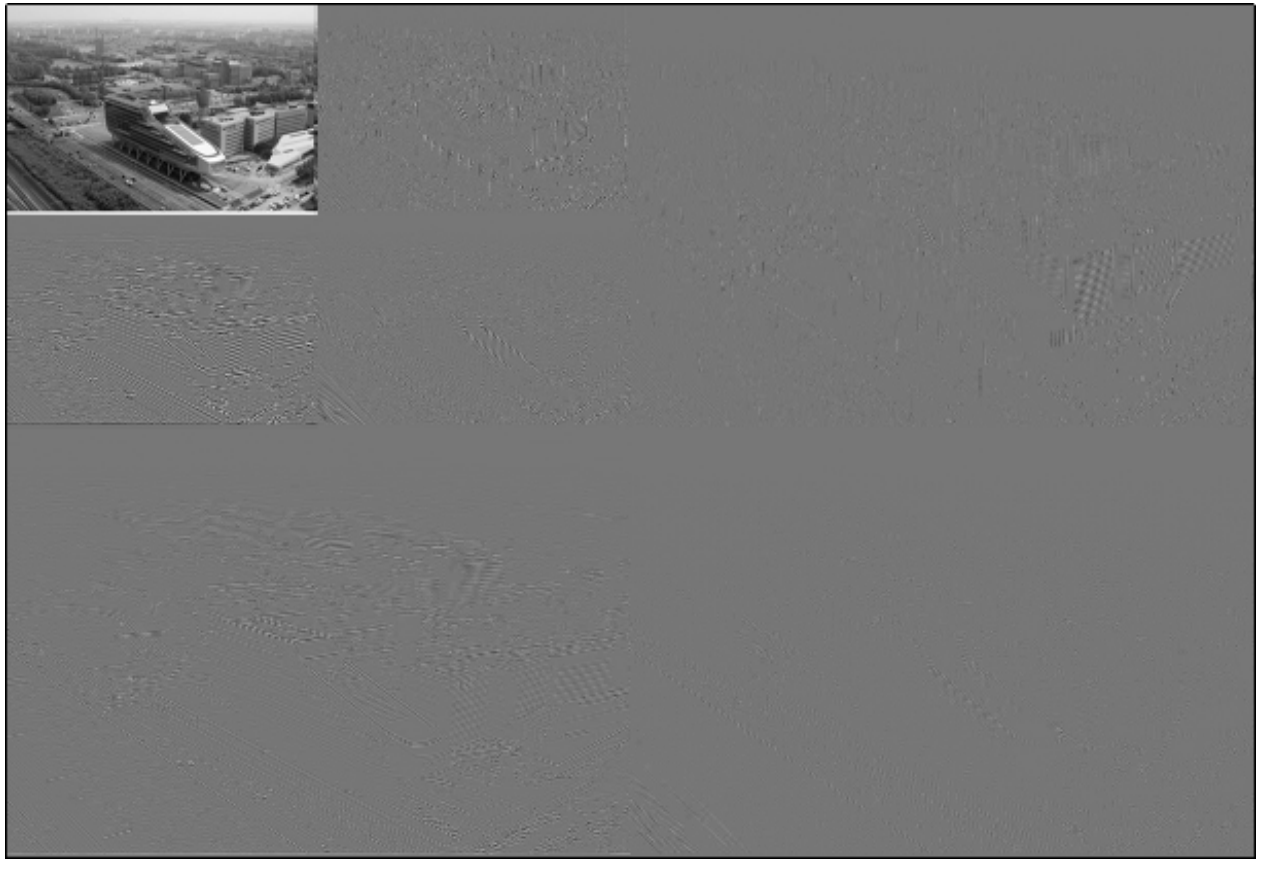
The matrix $\tilde{W}$ has the general shape

$$\begin{pmatrix}
-0.0238495 & -0.110624 & 0.377403 & 0.852699 & 0.377403 & -0.110624 & -0.0238495 & 0.0378285 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0378285 \\
0 & 0.0378285 & -0.0238495 & -0.110624 & 0.377403 & 0.852699 & 0.377403 & -0.110624 & -0.0238495 & 0.0378285 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.0378285 & -0.0238495 & -0.110624 & 0.377403 & 0.852699 & 0.377403 & -0.110624 & -0.0238495 & 0.0378285 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.0378285 & -0.0238495 & -0.110624 & 0.377403 & 0.852699 & 0.377403 & -0.110624 & -0.0238495 & 0.0378285 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0378285 & -0.0238495 & -0.110624 & 0.377403 & 0.852699 & 0.377403 & -0.110624 & -0.0238495 & 0.0378285 \\
-0.0238495 & 0.0378285 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0378285 & -0.0238495 & -0.110624 & 0.377403 & 0.852699 & 0.377403 & -0.110624 \\
0.377403 & -0.110624 & -0.0238495 & 0.0378285 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0378285 & -0.0238495 & -0.110624 & 0.377403 & 0.852699 \\
0.377403 & 0.852699 & 0.377403 & -0.110624 & -0.0238495 & 0.0378285 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0378285 & -0.0238495 & -0.110624 \\
0 & 0.0645389 & -0.0406894 & -0.418092 & 0.788486 & -0.418092 & -0.0406894 & 0.0645389 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.0645389 & -0.0406894 & -0.418092 & 0.788486 & -0.418092 & -0.0406894 & 0.0645389 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.0645389 & -0.0406894 & -0.418092 & 0.788486 & -0.418092 & -0.0406894 & 0.0645389 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0645389 & -0.0406894 & -0.418092 & 0.788486 & -0.418092 & -0.0406894 & 0.0645389 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0645389 & -0.0406894 & -0.418092 & 0.788486 & -0.418092 & -0.0406894 & 0.0645389 \\
-0.0406894 & 0.0645389 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0645389 & -0.0406894 & -0.418092 & 0.788486 & -0.418092 \\
0.788486 & -0.418092 & -0.0406894 & 0.0645389 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0645389 & -0.0406894 & -0.418092 \\
-0.0406894 & -0.418092 & 0.788486 & -0.418092 & -0.0406894 & 0.0645389 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0645389
\end{pmatrix}$$

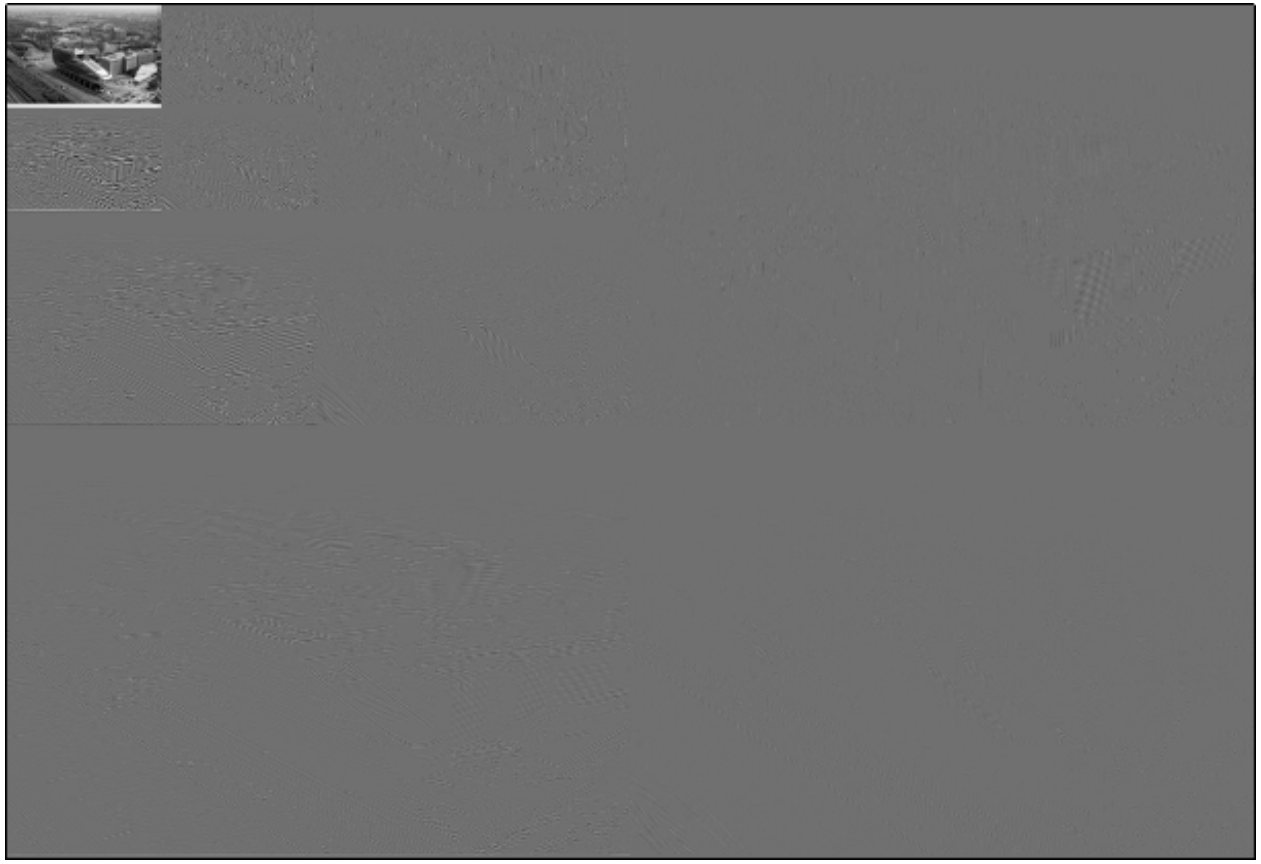
Given the matrix A for our digital image, we form $WA\tilde{W}^T$.



We then apply the same procedure to the upper left quarter of the matrix.



. . . and repeat one more time. (Entropy is 5.82.)



- The quantization step is similar to the quantization used in the JPEG algorithm. Once again thresholding and integer rounding is used with weights associated with the various regions of the transformed image. For the remainder we will use a quantized image with entropy 1.37. This is a compression factor of about 5.6.
- The resulting matrix is encoded and then sent to the receiver.
- The receiver "reverses" the quantization, then recovers the image using the orthogonality relation $\tilde{W}^T W = Id$.

Here is the recovered image:

The PSNR is about 29.7.

Comparison of JPEG and JPEG2000

- Similar performance for “normal” image compression

Comparison of JPEG and JPEG2000

- Similar performance for “normal” image compression
- JPEG is less computationally intensive.

Comparison of JPEG and JPEG2000

- Similar performance for “normal” image compression
- JPEG is less computationally intensive.
- Distribution of JPEG2000 hindered by a patent on the encoding algorithm.

Comparison of JPEG and JPEG2000

- Similar performance for “normal” image compression
- JPEG is less computationally intensive.
- Distribution of JPEG2000 hindered by a patent on the encoding algorithm.
- JPEG2000 performs better at larger compression factors; JPEG starts to produce blocking artifacts.

Comparison of JPEG and JPEG2000

- Similar performance for “normal” image compression
- JPEG is less computationally intensive.
- Distribution of JPEG2000 hindered by a patent on the encoding algorithm.
- JPEG2000 performs better at larger compression factors; JPEG starts to produce blocking artifacts.
- JPEG2000 permits a low-resolution analysis of images.

References

1. David Austin: *What is JPEG?* Notices of the AMS 55(2), 2008, pp. 226–9.
2. Catherine Beneteau, Patrick Van Fleet: *Discrete Wavelet Transformations and Undergraduate Education.* Notices of the AMS 58(5), 2011, pp. 656–66.
3. Patrick van Fleet: *Discrete Wavelet Transformations.* John Wiley & Sons, 2007.