Technische Universität München
Lehrstuhl für Informatik V
Univ.-Prof. Dr. M. Bader
M.Sc. A. Mo-Hellenbrand, M.Sc. A. Schwarz

SS 2016
Worksheet 3
4.5.2016

Algorithms of Scientific Computing

Discrete Cosine Transform – Solution

Exercise 1: Simple JPEG Encoder

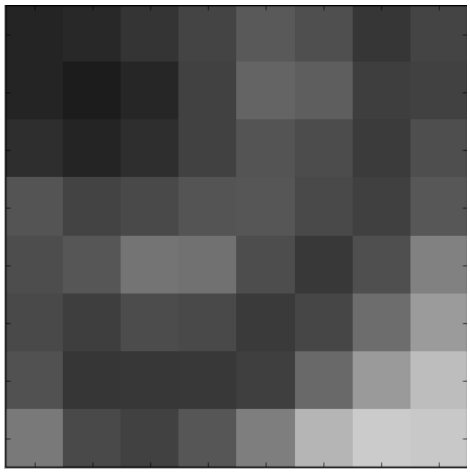
See the attached Python code. We created two lookup tables, one for the cosine values and one for the combination of coefficients. The DCT routine of the sample solution is very basic – the four nested loops take $\mathcal{O}(n^4)$.

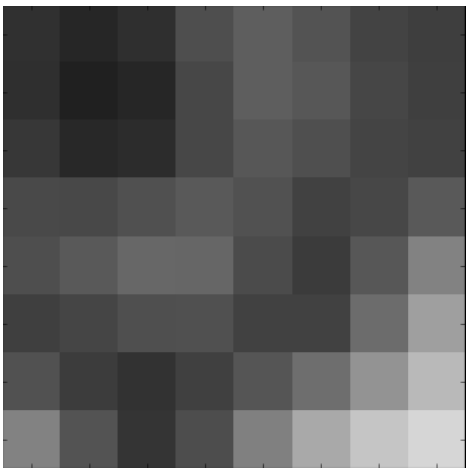
F_{00} is the average value. Speaking of images, this gives the background colour of an entire 8-by-8 block. Image processing refers to F_{00} as the DC coefficient. Coefficient F_{uv} far away from the DC coefficient contribute only minor information to the image. Quantisation reduces these coefficients.

The IDCT is straight forward. In contrast to the lecture, this IDCT utilises equally distributed normalisation factors. The IDCT therefore only switches the roles of pixels and frequencies.

$$f_{xy} = \frac{1}{\sqrt{2N}} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} c_u c_v F_{uv} \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right)$$

The original image is shown on the left; the one after encoding and decoding with JPEG is shown on the right.





The reconstructed image matches the original one pretty well, but appears smoother. Higher frequencies are quantised by the high values of the quantisation matrix, which results in a coarser approximation. The quantisation step of JPEG is nevertheless reasonable because the human visual system is less sensitive for high frequencies.

Exercise 2: Discrete Cosine Transform

a) Show that the corresponding Fourier coefficients are real

$$F_k = \frac{1}{2N} \sum_{n=-N+1}^{N} f_n \omega_{2N}^{-kn} \tag{1}$$

The proof is done in the following steps:

① Isolate the symmetry condition

② Insert the symmetry condition

③ Assemble terms to a sum over f_n

④ Make terms "real"

$$\begin{aligned}
F_k &= \frac{1}{2N} \sum_{n=-N+1}^{N} f_n \omega_{2N}^{-kn} \\
&\overset{①}{=} \frac{1}{2N} \left(\sum_{n=-N+1}^{-1} f_n \omega_{2N}^{-kn} + f_0 \omega_{2N}^{0} + \sum_{n=1}^{N-1} f_n \omega_{2N}^{-kn} + f_N \omega_{2N}^{-kN} \right) \\
&= \frac{1}{2N} \left(\sum_{n=1}^{N-1} f_{-n} \omega_{2N}^{kn} + f_0 e^{0} + \sum_{n=1}^{N-1} f_n \omega_{2N}^{-kn} + f_N e^{-i2\pi kN/2N} \right) \\
&\overset{②}{=} \frac{1}{2N} \left(\sum_{n=1}^{N-1} f_n \omega_{2N}^{kn} + f_0 + \sum_{n=1}^{N-1} f_n \omega_{2N}^{-kn} + f_N e^{-i\pi k} \right) \\
&\overset{③}{=} \frac{1}{2N} \left(f_0 + \sum_{n=1}^{N-1} f_n \underbrace{\left(\omega_{2N}^{kn} + \omega_{2N}^{-kn} \right)}_{= \omega_{2N}^{kn} + \left(\omega_{2N}^{kn}\right)^* = 2\mathrm{Re}\{\omega_{2N}^{kn}\}} + f_N e^{-i\pi k} \right) \\
&\overset{④}{=} \frac{1}{2N} \left(f_0 + 2 \sum_{n=1}^{N-1} f_n \underbrace{\mathrm{Re}\{e^{i2\pi kn/2N}\}}_{= \mathrm{Re}\{\cos\left(\frac{\pi kn}{N}\right) + i \sin\left(\frac{\pi kn}{N}\right)\}} + f_N \left(\cos(-\pi k) + i \sin(-\pi k)\right) \right) \\
&= \frac{1}{N} \left(\frac{1}{2} f_0 + \sum_{n=1}^{N-1} f_n \cos\left(\frac{\pi kn}{N}\right) + \frac{1}{2} f_N \cos(\pi k) \right) \in \mathbb{R} \qquad \text{q.e.d.}
\end{aligned}$$

b) Show that the F_k also have a symmetry:

Due to $\cos(x) = \cos(-x)$ we obtain easily:

$$\begin{aligned} F_{-k} &= \frac{1}{N}\left(\frac{1}{2}f_0 + \sum_{n=1}^{N-1} f_n \cos\left(\frac{-\pi kn}{N}\right) + \frac{1}{2}f_N \cos(-\pi k)\right) \\ &= \frac{1}{N}\left(\frac{1}{2}f_0 + \sum_{n=1}^{N-1} f_n \cos\left(\frac{\pi kn}{N}\right) + \frac{1}{2}f_N \cos(\pi k)\right) \\ &= F_k \end{aligned}$$

Since all $F_{-k} = F_k$, we need the F_k only for $k = 0, \ldots, N$ for a Cosine Transform.

c) Algorithm for the Cosine Transform

The procedure `FFT(f,N)` computes the correct coefficients, if we pass the $N+1$ data from field `g` as a dataset of length $2N$ with symmetry $f_{-n} = f_n$.

From equation (1) of the worksheet we know that `FFT(f,N)` gets a dataset `f` with indices $n = -N+1, \ldots, N$. We only have to compute the F_k for $k = 0, \ldots, N$.

So, the algorithm looks like this:

1. Set `f[0] := g[0]` $= f_0$
   For all $n = 1, \ldots, N-1$:
   Set `f[n] := g[n]` $= f_n$
   Set `f[-n] := g[n]` $= f_n$
   Set `f[N] := g[N]` $= f_N$
2. Call `FFT(f,N)`
3. *(Now the Fourier coefficients F_k are stored in the field `f`)*
   For all $k = 0, \ldots, N$:
   Set `g[k] := f[k]` $= F_k$

Exercise 3: Fast Discrete Cosine Transform

The butterfly scheme is retrieved as usual:

$$\begin{aligned} F_k &= \frac{1}{2N}\sum_{n=-N+1}^{N} f_n \omega_{2N}^{-kn} = \frac{1}{2}\left(\frac{1}{N}\sum_{n=-N/2+1}^{N/2} f_{2n}\omega_{2N}^{-2kn} + \frac{1}{N}\sum_{n=-N/2+1}^{N/2} f_{2n-1}\omega_{2N}^{-k(2n-1)}\right) \\ &= \frac{1}{2}\Bigg(\underbrace{\frac{1}{N}\sum_{n=-N/2+1}^{N/2} f_{2n}\omega_N^{-kn}}_{=:G_k} + \underbrace{\frac{1}{N}\sum_{n=-N/2+1}^{N/2} f_{2n-1}\omega_N^{-kn}}_{=:H_k}\,\omega_{2N}^{k}\Bigg) \\ &= \frac{1}{2}\left(G_k + \omega_{2N}^{k} H_k\right) \\ F_{k+N} &= \frac{1}{2}\left(G_{k+N} + \omega_{2N}^{k+N} H_{k+N}\right) = \frac{1}{2}\left(G_k - \omega_{2N}^{k} H_k\right) \end{aligned}$$

For the datasets $g_n := f_{2n}$ and $h_n := f_{2n-1}$, respectively, we can try to find other symmetries:

$$g_{-n} = f_{2(-n)} = f_{-2n} = f_{2n} = g_n$$

The "even" data also shows an even symmetry and therefore leads to another Cosine Transform but with half length.

Analogously, for the data with odd indices:

$$h_{-n} = f_{2(-n)-1} = f_{-2n-1} = f_{2n+1} = f_{2(n+1)-1} = h_{n+1}$$

Again we get an "even" symmetry. This is the transform shown in the lecture, known as Quarter-Wave-DCT, again with half length.

For a dataset with the symmetry constraint $f_{-n} = f_{n+1}$ we get accordingly

$$g_{-n} = f_{2(-n)} = f_{-2n} = f_{2n+1} = h_{n+1}$$

and

$$h_{-n} = f_{-2n-1} = f_{-2n+1} = f_{2n+2} = f_{2n-1} = g_{n+1}$$