Solutions: 4F8 2015

ENGINEERING TRIPOS PART IIB

Friday 1st May 2015 2 to 3.30

Module 4F8

IMAGE PROCESSING AND IMAGE CODING

Version: NGK/3

1 (a) (i) First take the FT of the central stripe, call this $S1$:

$$\begin{aligned} G_{S1}(\omega_1,\omega_2) &= \int_{-b}^{+b}\int_{-a/2}^{+a/2} A\mathrm{e}^{-j\omega_1 u_1}\mathrm{e}^{-j\omega_2 u_2}\,du_1 du_2 \\ &= A\int_{-b}^{+b}\mathrm{e}^{-j\omega_2 u_2}du_2\int_{-a/2}^{+a/2}\mathrm{e}^{-j\omega_1 u_1}du_1 \\ &= A\left[\frac{\mathrm{e}^{-j\omega_1 u_1}}{-j\omega_1}\right]_{-a/2}^{a/2}\left[\frac{\mathrm{e}^{-j\omega_2 u_2}}{-j\omega_2}\right]_{-b}^{b} \\ &= 2Aab\ \mathrm{sinc}\frac{a\omega_1}{2}\ \mathrm{sinc}\,b\omega_2 \end{aligned}$$

...or just use standard result for the FT of a rectangular pulse from the Databook.

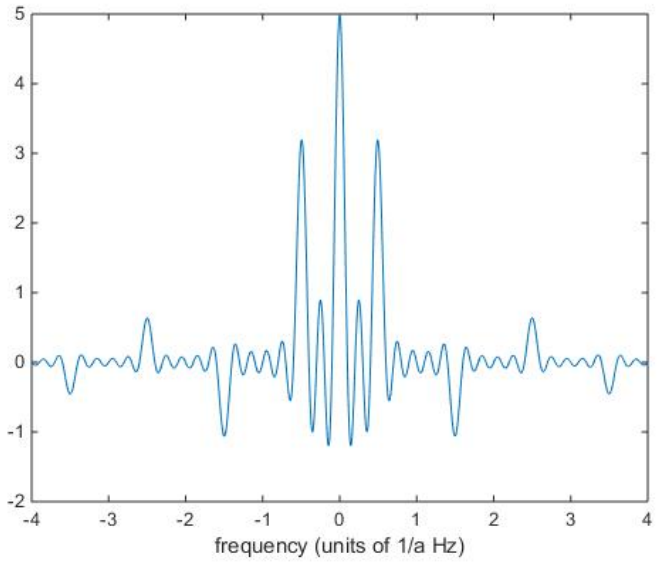
Now, each of the other 4 dark stripes are just shifted versions of this, so their FT will just be the above multiplied by the appropriate phase factor. If from left to right the stripes are $S5, S3, S1, S2, S4$, the FTs are (using $g(u_1-\mu_1,u_2-\mu_2) \Leftrightarrow \mathrm{e}^{-j(\mu_1\omega_1+\mu_2\omega_2)}\,G_{S1}(\omega_1,\omega_2)$):

$$G_{S2}=G_{S1}\mathrm{e}^{-j2a\omega_1},\ G_{S3}=G_{S1}\mathrm{e}^{j2a\omega_1},\ G_{S4}=G_{S1}\mathrm{e}^{-j4a\omega_1},\ G_{S5}=G_{S1}\mathrm{e}^{j4a\omega_1}$$

The total FT G is therefore the sum of these 5 FTs:

$$\begin{aligned} G(\omega_1,\omega_2) &= G_{S1}\left[\mathrm{e}^{j4a\omega_1}+\mathrm{e}^{j2a\omega_1}+1+\mathrm{e}^{-j2a\omega_1}+\mathrm{e}^{-j4a\omega_1}\right] \\ &= 2Aab\ \mathrm{sinc}\,b\omega_2\ \mathrm{sinc}\frac{a\omega_1}{2}\ [1+2\cos 2a\omega_1+2\cos 4a\omega_1] \end{aligned}$$

This is a pure sinc function along the ω_2 axis with a single main-lobe peak at $\omega_2=0$, and zeros at multiples of $\pi/b=2\pi/9a$ on either side of the main lobe. This is multiplied by a modulated sinc function along the ω_1 axis, such that the sinc function has a main lobe peak at $\omega_1=0$, and zeros at multiples of $2\pi/a$ on either side of the main lobe. The modulating function $[1+2\cos 2a\omega_1+2\cos 4a\omega_1]$ has sharp positive peaks of value 5 at multiples of π/a, where all 3 terms add coherently, and small ripples in between these peaks. The even-numbered sharp peaks are suppressed by the zeros of the $\mathrm{sinc}\frac{a\omega_1}{2}$ function (except at $\omega_1=0$), and the odd-numbered sharp peaks coincide with the centres of the lobes of the sinc function. This is illustrated along the ω_1 axis in the figure below:



The spectrum in the ω_2 direction is a sinc function, consistent with a waveform that is a single rectangular pulse of width $2b$. The spectrum in the ω_1 direction is approximately a set of impulses, consistent with the spectrum of an infinitely long square wave of period $2a$, convolved with a sinc function that is consistent with multiplication in the spatial domain by a pulse of width $9a$.

[40%]

(ii) To avoid aliasing we need to sample at more than twice the largest (significant) frequency in the image. But the spectrum is composed of sinc functions along both frequency axes so we must make assumptions / approximations. The spectrum is wider in the ω_1 direction so we assume that, in order to preserve reasonably sharp edges, we must pass frequencies up to (say) the 6th harmonic of the square wave fundamental frequency $1/2a$ Hz; i.e. up to $\pm 3/a$ Hz, which includes the peaks at $1.5/a$ and $2.5/a$ in the above figure. Hence the sampling frequency should be at least twice this, which is $6/a$; i.e. the sampling period should be $\leq a/6$.

A lower sampling frequency could be used in the ω_2 direction but it is usually better to sample images at the same rate in both directions so that the amount of blurring appears the same in all directions. [10%]

(b) (i) The Product Method for obtaining a 2-D window from 1-D windows is to simply take the product of two 1-D windows:

$$\boxed{w(n_1,n_2) = w_1(n_1)\, w_2(n_2)}$$

The Rotation Method of forming a 2-D window from 1-D windows is to obtain a 2-D *continuous* window function $w(u_1,u_2)$ by rotating a 1-D continuous window $w_1(u)$.

$$\boxed{w(u_1,u_2) = w_1(u)\big|_{u=\sqrt{(u_1^2+u_2^2)}}}$$

The continuous 2-D window is then sampled to produce a discrete 2-D window $w(n_1,n_2)$:

$$\boxed{w(n_1,n_2) = w(u_1,u_2)|_{u_1=n_1\Delta_1,\, u_2=n_2\Delta_2}}$$

The actual filter frequency response $H(\omega_1,\omega_2)$ is given by the **convolution** of the desired frequency response $H_d(\omega_1,\omega_2)$ with the window function spectrum $W(\omega_1,\omega_2)$.

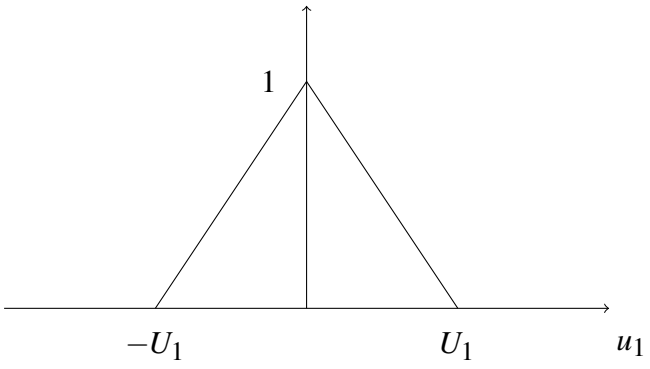
Thus the effect of the window is to smooth H_d; so clearly we would prefer to have the mainlobe width of $W(\omega_1,\omega_2)$ small so that H_d is changed as little as possible. We also want its sidelobes to be of small amplitude so that ripples in the (ω_1,ω_2) plane outside the region of interest are kept small.

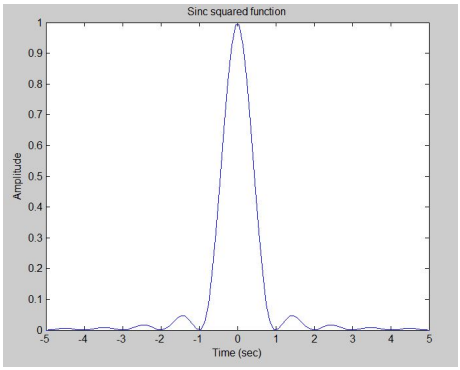
[15%]

(ii) This is a triangular window of width $2U$ and of unit height. From the E&I databook, its FT is given by:

$$U_1 \,\mathrm{sinc}^2 \frac{\omega_1 U_1}{2}$$

Sketches of w_1 and W_1 are given below. On the sinc^2 plot the proper amplitude (U_1) should be marked and the positions of the first nulls ($\pm 2\pi/U_1$).





The full 2D spectrum is therefore given by the product of the separate 1D spectra:

$$W(\omega_1, \omega_2) = W_1(\omega_1)\, W_2(\omega_2) = U_1 U_2 \ \text{sinc}^2 \frac{\omega_1 U_1}{2} \ \ \text{sinc}^2 \frac{\omega_2 U_2}{2}$$

[25%]

(iii) As the triangular window produces a sinc^2 spectrum with its first nulls at $\omega_i = 2\pi/U_i$, the mainlobe width decreases as U_i increases and the sidelobes drop down in magnitude fairly quickly – but there is no suppression of sidelobes. A smoother window function such as a raised cosine (Hamming or Hanning window) would offer lower amplitude (and possibly faster decaying) sidelobes with about the same main-lobe width.

[10%]

2 (a) (i) We assume that our observed image, $y(\mathbf{n})$, can be modelled as a convolution of the true image, $x(\mathbf{n})$, with a point spread function (psf), $h(\mathbf{n})$, plus additive noise, $d(\mathbf{n})$, i.e.

$$y(\mathbf{n}) = \sum_{\mathbf{m}\in\mathbb{Z}^2} h(\mathbf{m})\, x(\mathbf{n}-\mathbf{m}) + d(\mathbf{n})$$

Here we have used vector notation to denote the locations of the discrete pixels of the image; e.g. $\mathbf{n} = (n_1, n_2)$ and $\mathbf{m} = (m_1, m_2)$. [10%]

(ii) If we ignore noise, we can take the Fourier transform of each side of the above convolution equation to give:

$$Y(\omega_1, \omega_2) = H(\omega_1, \omega_2) X(\omega_1, \omega_2)$$

$$\therefore \;\; X(\omega_1, \omega_2) = \frac{Y(\omega_1, \omega_2)}{H(\omega_1, \omega_2)}$$

The $1/H$ term is known as the *inverse filter*. If $H(\omega_1, \omega_2)$ has zeros, then the inverse filter, $1/H$, will have infinite gain. If $1/H$ is very large (or indeed infinite), small noise in the regions of the frequency plane where these large values of $1/H$ occur will be hugely amplified. To counter this we can threshold the frequency response, leading to the so-called, pseudo-inverse or generalised inverse filter $H_g(\omega_1, \omega_2)$ given by

$$H_g(\omega_1, \omega_2) = \begin{cases} \frac{1}{H(\omega_1,\omega_2)} & \frac{1}{|H(\omega_1,\omega_2)|} < \gamma \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

or

$$H_g(\omega_1, \omega_2) = \begin{cases} \frac{1}{H(\omega_1,\omega_2)} & \frac{1}{|H(\omega_1,\omega_2)|} < \gamma \\ \gamma \frac{|H(\omega_1,\omega_2)|}{H(\omega_1,\omega_2)} & \text{otherwise} \end{cases} \qquad (2)$$

Clearly for $\frac{1}{|H(\omega_1,\omega_2)|} \geq \gamma$ in equation 2, the modulus of the filter is set as γ, whereas in the previous equation it is set as 0.

Although the *generalised inverse filter* may perform reasonably well on *noiseless images*, the performance is unsatisfactory with even mildly noisy images due to the still significant noise gain at frequencies where $|H(\omega_1, \omega_2)|$ is relatively small. [10%]

(iii) (Note: this is a very full solution from the lecture notes. Shorter versions with fewer intermediate steps would be acceptable.)
To obtain the standard form of the Wiener filter (in terms of P_{xx}, P_{dd} and H) we firstly look at the autocorrelation function of y:

$$R_{yy}(\mathbf{p}) = E\{y(\mathbf{n})\, y(\mathbf{n}-\mathbf{p})\} \text{ where } y(\mathbf{n}) = \sum_{\mathbf{m}} h(\mathbf{m})\, x(\mathbf{n}-\mathbf{m}) + d(\mathbf{n})$$

We assume that the images are spatially statistically stationary and that x and y are real. If signal and noise are uncorrelated and noise is zero mean:

$$R_{yy}(\mathbf{p}) = E\left\{\sum_{\mathbf{m}}\sum_{\mathbf{q}} h(\mathbf{m})\, x(\mathbf{n}-\mathbf{m})\, h(\mathbf{q})\, x(\mathbf{n}-\mathbf{p}-\mathbf{q})\right\} + E\left\{d(\mathbf{n})\, d(\mathbf{n}-\mathbf{p})\right\}$$

$$= \sum_{\mathbf{m}}\sum_{\mathbf{q}} h(\mathbf{m})\, h(\mathbf{q}) E\left\{x(\mathbf{n}-\mathbf{m})\, x(\mathbf{n}-\mathbf{p}-\mathbf{q})\right\} \; + \; R_{dd}(\mathbf{p})$$

$$\therefore \;\; R_{yy}(\mathbf{p}) = \sum_{\mathbf{m}}\sum_{\mathbf{q}} h(\mathbf{m})\, h(\mathbf{q})\, R_{xx}(\mathbf{p}+\mathbf{q}-\mathbf{m}) \; + \; R_{dd}(\mathbf{p})$$

Now take the Fourier transform of each side to give:

$$P_{yy}(\omega) = \sum_{\mathbf{p}} \left\{\sum_{\mathbf{m}}\sum_{\mathbf{q}} h(\mathbf{m})\, h(\mathbf{q})\, R_{xx}(\mathbf{p}+\mathbf{q}-\mathbf{m})\right\} e^{-j\omega^T \mathbf{p}} \; + \; P_{dd}(\omega)$$

where P_{dd} is the FT of the autocorrelation function of the noise. Interchange order:

$$P_{yy}(\omega) = \sum_{\mathbf{m}}\sum_{\mathbf{q}} h(\mathbf{m})\, h(\mathbf{q}) \sum_{\mathbf{p}} R_{xx}(\mathbf{p}+\mathbf{q}-\mathbf{m})\, e^{-j\omega^T \mathbf{p}} \; + \; P_{dd}(\omega)$$

Let $\mathbf{k} = (\mathbf{p}+\mathbf{q}-\mathbf{m})$, then:

$$P_{yy}(\omega) = \sum_{\mathbf{m}}\sum_{\mathbf{q}} h(\mathbf{m})\, h(\mathbf{q}) \sum_{\mathbf{k}} R_{xx}(\mathbf{k})\, e^{-j\omega^T (\mathbf{k}-\mathbf{q}+\mathbf{m})} \; + \; P_{dd}(\omega)$$

$$= \left\{\sum_{\mathbf{m}} h(\mathbf{m})\, e^{-j\omega^T \mathbf{m}}\right\} \left\{\sum_{\mathbf{q}} h(\mathbf{q})\, e^{j\omega^T \mathbf{q}}\right\} \left\{\sum_{\mathbf{k}} R_{xx}(\mathbf{k})\, e^{-j\omega^T \mathbf{k}}\right\} \; + \; P_{dd}(\omega)$$

$$\therefore \;\; P_{yy}(\omega) = |H(\omega)|^2\, P_{xx}(\omega) \; + \; P_{dd}(\omega) \qquad (3)$$

as h is real.

Now look at the cross correlation function of x and y:

$$R_{xy}(\mathbf{p}) = E\{x(\mathbf{n})\, y(\mathbf{n}-\mathbf{p})\}$$

$$= E\left\{\left[\sum_{\mathbf{m}} h(\mathbf{m})\, x(\mathbf{n}-\mathbf{p}-\mathbf{m}) + d(\mathbf{n})\right] x(\mathbf{n})\right\}$$

The image $x(\mathbf{n})$ and the noise $d(\mathbf{n})$ are uncorrelated and the noise has zero mean (as before), so $E\{d(\mathbf{n})\, x(\mathbf{n})\} = 0$.

$$\therefore \;\; R_{xy}(\mathbf{p}) = E\left\{\sum_{\mathbf{m}} h(\mathbf{m})\, x(\mathbf{n}-\mathbf{p}-\mathbf{m})\, x(\mathbf{n})\right\}$$

$$= \sum_{\mathbf{m}} h(\mathbf{m}) E\{x(\mathbf{n})\, x(\mathbf{n}-[\mathbf{p}+\mathbf{m}])\} = \sum_{\mathbf{m}} h(\mathbf{m})\, R_{xx}(\mathbf{p}+\mathbf{m})$$

Taking the Fourier transform of each side gives:

$$P_{xy}(\omega) = \sum_{\mathbf{p}} \left\{\sum_{\mathbf{m}} h(\mathbf{m})\, R_{xx}(\mathbf{p}+\mathbf{m})\right\} e^{-j\omega^T \mathbf{p}} = \sum_{\mathbf{m}} h(\mathbf{m}) \sum_{\mathbf{p}} R_{xx}(\mathbf{p}+\mathbf{m})\, e^{-j\omega^T \mathbf{p}}$$

$$= \sum_{\mathbf{m}} h(\mathbf{m})\, e^{j\omega^T \mathbf{m}} \sum_{\mathbf{k}} R_{xx}(\mathbf{k})\, e^{-j\omega^T \mathbf{k}} \qquad \text{where } \; \mathbf{p}+\mathbf{m} = \mathbf{k}$$

$$\therefore \;\; P_{xy}(\omega) = H^*(\omega)\; P_{xx}(\omega)$$

Substituting back into the given equation gives:

$$\boxed{G(\omega) = \frac{H^*(\omega)\, P_{xx}(\omega)}{|H(\omega)|^2\, P_{xx}(\omega) + P_{dd}(\omega)}} \qquad (4)$$

This is the most commonly used form of the Wiener Filter. [30%]

(b) (i) Consider the ideal filter given in fig.2 – one way to construct this is to say that the ideal frequency response of this filter, $H(\omega_1, \omega_2)$, can be written as

$$H(\omega_1, \omega_2) = H_1(\omega_1, \omega_2) + H_2(\omega_1, \omega_2) + H_3(\omega_1, \omega_2)$$

where H_1 is a rectangular 2D lowpass filter given by

$$H_1(\omega_1, \omega_2) = \begin{cases} 1 & \text{if } |\omega_1| < \Omega \text{ and } |\omega_2| < \Omega \\ 0 & \text{otherwise} \end{cases}$$

and H_2 and H_3 are ideal 2D bandpass filters. H_2 is formed from the product of 1D bandpass filters in ω_1 and ω_2, which are each obtained by subtracting a 1D lowpass filter of bandwidth Ω from one of bandwidth 3Ω. Similarly H_3 is formed from the product of 1D filters, obtained by subtracting a lowpass filter of bandwidth 3Ω from one of bandwidth 5Ω.

We can therefore use standard results (or derive them) to write our impulse response of H as the sum of the impulse responses of H_1, H_2 and H_3 :

$$h(n_1\Delta_1, n_2\Delta_2) = \frac{\Delta_1\Delta_2}{\pi^2}[\Omega^2 \operatorname{sinc}(\Omega n_2\Delta_2)\operatorname{sinc}(\Omega n_1\Delta_1)]$$
$$+\frac{\Delta_1\Delta_2}{\pi^2}[3\Omega\operatorname{sinc}(3\Omega n_1\Delta_1)) - \Omega\operatorname{sinc}(\Omega n_1\Delta_1))] \times$$
$$[3\Omega\operatorname{sinc}(3\Omega n_2\Delta_2) - \Omega\operatorname{sinc}(\Omega n_2\Delta_2)]$$
$$+\frac{\Delta_1\Delta_2}{\pi^2}[5\Omega\operatorname{sinc}(5\Omega n_1\Delta_1)) - 3\Omega\operatorname{sinc}(3\Omega n_1\Delta_1))] \times$$
$$[5\Omega\operatorname{sinc}(5\Omega n_2\Delta_2) - 3\Omega\operatorname{sinc}(3\Omega n_2\Delta_2)]$$

If we expand this we have

$$h(n_1\Delta_1, n_2\Delta_2) = \frac{\Delta_1\Delta_2}{\pi^2}\Omega^2[2\operatorname{sinc}(\Omega n_2\Delta_2)\operatorname{sinc}(\Omega n_1\Delta_1)$$
$$+ 18\operatorname{sinc}(3\Omega n_2\Delta_2)\operatorname{sinc}(3\Omega n_1\Delta_1)$$
$$+ 25\operatorname{sinc}(5\Omega n_2\Delta_2)\operatorname{sinc}(5\Omega n_1\Delta_1)$$
$$+ \text{other terms}]$$

Thus the values required are $\alpha_0 = 2,\ \alpha_1 = 18,\ \alpha_2 = 25.$ [40%]

(ii) Clearly the filter only lets 'diagonal' frequencies through, and also low frequency components which lie within the passband of the central square region of the response. Thus we can expect this filter to pick out parts of the image with strong 'diagonal' features – ie patterns which exhibit frequencies which are simultaneously similar in both directions – as well as a generally blurred (lowpass) version of the image.

There will be quite a lot of *ringing* around edges due to the sharp transitions of the frequency response in Fig. 2. [10%]

3 (a) In RGB space, colour (chrominance) and intensity (luminance) information is spread across all 3 components, R, G and B. The transformation to YUV space aims to separate the information into the luminance component, Y, and 2 chrominance components, U and V. The top row of $\mathbf{C}$ combines $0.3R+0.6G+0.1B$ to give Y, as these are the correct ratios to give a subjectively correct estimate of the apparent brightness (luminance) of each pixel. They should add to unity so that $Y = R = G = B$ for colourless pixels of any shade of grey from black (0,0,0) to white (255,255,255) in RGB space.

The 2nd and 3rd rows of $\mathbf{C}$ must sum to zero, so that $U = V = 0$ when $R = G = B$ and the pixel is colourless. We find that $U \propto B - Y$ and $V \propto R - Y$. Thus U and V are known as colour-difference components in the blue and red directions.

The human eye is much more sensitive to high frequency components in Y than it is to high frequencies in U and V; so, in an image coder, U and V can be lowpass filtered and sub-sampled without producing any noticeable degradation of image quality. This saves significant bit rate when colour images are encoded. [20%]

(b) If $\mathbf{T}$ is a 4×4 DCT matrix which transforms a 4-element column vector $\mathbf{x}$ into a 4-vector $\mathbf{y}$ of transform coefficients, then $\mathbf{y} = \mathbf{T}\mathbf{x}$. To transform both the rows and columns of a 4×4 patch of pixels $\mathbf{X}$, we use pre and post multiplication by $\mathbf{T}$ and $\mathbf{T}^T$ to obtain

$$\hat{\mathbf{X}} = \mathbf{T}\,\mathbf{X}\,\mathbf{T}^T \tag{1}$$

where $\hat{\mathbf{X}}$ is a 4×4 matrix of DCT coefficients.

To transform a large image, we divide the image into non-overlapping 4×4 blocks of pixels, and apply eq(1) to each block in turn.

To view the result we usually group equivalent pixels from all the different blocks $\hat{\mathbf{X}}$ into $4 \times 4 = 16$ separate subimages. Each subimage is $\frac{1}{4}$ of the size of the original image in each direction. Hence an $N \times M$ input image produces 16 subimages, each of size $\frac{N}{4} \times \frac{M}{4}$ pixels, with one subimage corresponding to each of the 16 coefficient locations in $\hat{\mathbf{X}}$. [20%]

(c) The Y image will be 2048×1536 in size, and the U' and V' images will each be 1024×768 in size. The DCT subimages will be $\frac{1}{4}$ of these dimensions in each direction, so the 16 subimages of Y will each be of size 512×384, and the 32 submiages of U' and V' will each be of size 256×192 .

As i and j go from 1 to 4, the sets of images on each diagonal of the 4×4 grid, such that $i + j - 1 =$ constant, will have the same approximate entropy. Hence we can form a

table of entropies as follows:

Diagonal $i+j-1$	No. of subimages on the diagonal m	H_Y	$m\,H_Y$	H_{UV}	$m\,H_{UV}$
1	1	6	6	6	6
2	2	3	6	1.5	3
3	3	2	6	0.667	2
4	4	1.5	6	0.375	1.5
5	3	1.2	3.6	0.24	0.72
6	2	1	2	0.167	0.333
7	1	0.857	0.857	0.122	0.122

For the luminance plane Y, assuming that the mean bit rate per coefficient is equal to the entropy of that subband, the total no. of bits will be:

$$N_Y = 512 \times 384 \times (\sum mH_Y) = 512 \times 384 \times (4 \times 6 + 3.6 + 2 + 0.857)$$

$$= 512 \times 384 \times 30.46 = 5.989 \;.\; 10^6 \text{ bits}$$

For U and V, it will be:

$$N_{UV} = 2 \times 256 \times 192 \times (\sum mH_{UV}) = 512 \times 192 \times (6 + 3 + 2 + 1.5 + 0.72 + 0.333 + 0.122)$$

$$= 512 \times 192 \times 13.675 = 1.344 \;.\; 10^6 \text{ bits}$$

Hence we need approx $5.99 \;.\; 10^6$ bits to code the luminance and approx $1.34 \;.\; 10^6$ bits for the two chrominance planes together.

The total number of bits to code each colour image will therefore be approx [40%]

$$N_Y + N_{UV} = (5.99 + 1.34) \;.\; 10^6 = 7.33 \;.\; 10^6 \text{ bits}$$

(d) We see that both colour components together only add about 23% to the bit-rate for the luminance component. So it is usually well worth encoding the image in colour, since it looks so much better and more interesting than an equivalent monochrome image.

The low cost of colour is to be expected from knowledge of the human visual system, because the *human eye* is much less sensitive to colour information, both in terms of bandwidth and contrast sensitivity. (See fig 1.1 in the 4F8 lecture notes.) [20%]

4 (a) **y = T x** transforms a 2-element column vector of pixels **x** into a 2-element coefficient vector **y** as a simple sum and difference operation, scaled by $\frac{1}{\sqrt{2}}$ to preserve energy.

To transform the rows *and* columns of a 2×2 matrix **X**, we calculate

$$\mathbf{Y} = \mathbf{T}\,\mathbf{X}\,\mathbf{T}^T$$

If we have the inverse of **T** as matrix $\mathbf{T}^{-1}$, then we can recover **X** from **Y** using

$$\mathbf{T}^{-1}\,\mathbf{Y}\,(\mathbf{T}^{-1})^T = \mathbf{T}^{-1}\,\mathbf{T}\,\mathbf{X}\,\mathbf{T}^T\,(\mathbf{T}^{-1})^T = \mathbf{I}\,\mathbf{X}\,\mathbf{I} = \mathbf{X}$$

The important property of **T** is that it is *orthonormal* (i.e. orthogonal and normalized for unit energy along rows or columns). The inverse of an orthonormal matrix is just its transpose, so that $\mathbf{T}^{-1} = \mathbf{T}^T$ and $\mathbf{T}^T\,\mathbf{T} = \mathbf{I}$. Orthonormal transformations also preserve energy.

Hence to recover **X** from **Y**, we just calculate

$$\mathbf{T}^T\,\mathbf{Y}\,\mathbf{T} = \mathbf{T}^T\,\mathbf{T}\,\mathbf{X}\,\mathbf{T}^T\,\mathbf{T} = \mathbf{X}$$

This only requires simple sum and difference operations on **Y**, followed by an overall scaling by $\frac{1}{2}$. [15%]

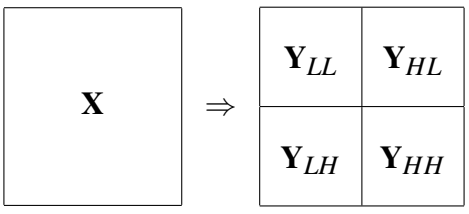
(b) **Y** is a 2×2 matrix of transform coefs.

$$\text{If } \mathbf{X} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad \text{then } \mathbf{Y} = \frac{1}{2}\begin{bmatrix} a+b+c+d & a-b+c-d \\ a+b-c-d & a-b-c+d \end{bmatrix}$$

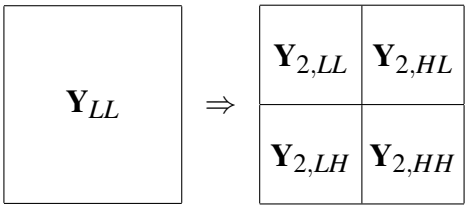
This is equivalent to 4 filters

$$\begin{bmatrix} \text{Lo-Lo} & \text{Hi-Lo} \\ \text{Lo-Hi} & \text{Hi-Hi} \end{bmatrix} \qquad \text{where} \quad \text{Lo = lowpass} \quad \& \quad \text{Hi = highpass}$$

The 4 elements of all the different **Y** matrices for a large image can be split into 4 separate image subbands:



$\mathbf{Y}_{HL}$, $\mathbf{Y}_{LH}$, $\mathbf{Y}_{HH}$ all result from highpass filtering and contain much less energy than $\mathbf{Y}_{LL}$. In a multi-level transform, better energy compression is obtained if the LL subband is passed through one or more further stages of Haar transform. Hence we get



and this can be continued for several stages, using $\mathbf{Y}_{2,LL}$ as the input for the next stage to obtain $\mathbf{Y}_{3,LL}$ ... $\mathbf{Y}_{3,HH}$, and so on ...

At each stage (level) the Lo-Lo subband is subdivided into 4 further subbands, each half the size (in each direction) of the input band for that stage.

Hence we see that after 4 levels, there are 3 bandpass subbands for each level plus a final Lo-Lo band, giving a total of $4 \times 3 + 1 = 13$ subbands.

If the input image is $N \times N$, then there will be 3 subbands of size $\frac{N}{2} \times \frac{N}{2}$, 3 subbands $\frac{N}{4} \times \frac{N}{4}$, 3 subbands $\frac{N}{8} \times \frac{N}{8}$, and 4 subbands $\frac{N}{16} \times \frac{N}{16}$. These all add up to $N \times N$ coefficients, which equals the number of input pixels and shows that the transform is non-redundant. [20%]

(c) When the Lo-Lo filters are cascaded over m levels, they simply produce coefs. that are the scaled average of a $2^m \times 2^m$ region of pixels. This produces a filter whose point-spread function (PSF) is a $2^m \times 2^m$ square of uniform value, with abrupt transitions to zero at the edges of the square. When a transformed image is encoded at a low bit-rate by quantizing coefs., many of the smaller coefs. are set to zero. The reconstructed image is then made up of scaled filter PSFs from only the sparse non-zero coefs. Hence if these PSFs are square blocks, the reconstructed image will appear *blocky*.

To improve on the Haar transform, we observe that its sum and difference operations can be interpreted as filtering operations where

$$H_0(z) = \frac{1}{\sqrt{2}}(1+z^{-1}) \quad \text{and} \quad H_1(z) = \frac{1}{\sqrt{2}}(1-z^{-1})$$

Wavelet transforms aim to overcome the problem of blocky artifacts by extending these filters to higher order (more terms) so that, when cascaded across scales or levels, they produce smoothly decaying PSFs with less abrupt boundaries. [25%]

(d) To calculate the filter products, we multiply their polynomials in z. This can be done more easily as a discrete convolution (noting that a polynomial in z^2 spaces the terms 2 samples apart).

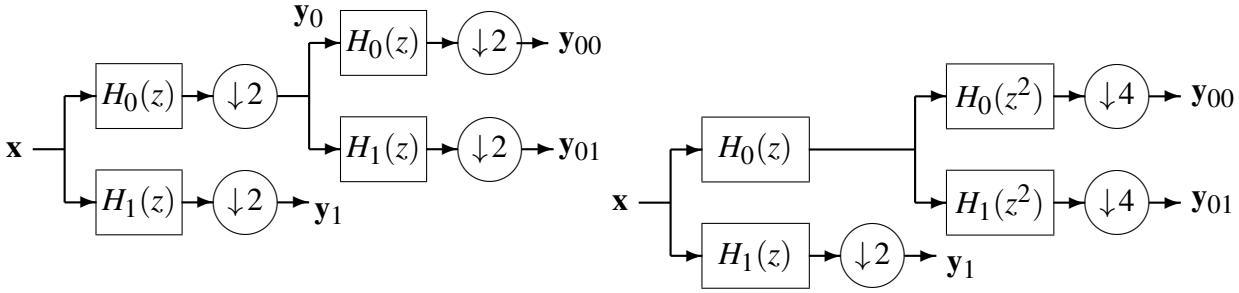
$H_0(z) \times H_1(z^2) \times 16$:

$-z^0$:	1	-2	-6	-2	1				
$2z^{-2}$:			-2	4	12	4	-2		
$-z^{-4}$:					1	-2	-6	-2	1
Sum:	1	-2	-8	2	14	2	-8	-2	1

Similarly $G_0(z) \times G_1(z^2) \times 16$ gives:

Sum:	-1	-2	-3	-4	4	12	4	-4	-3	-2	-1

To explain why these products represent level-2 basis functions, we note that the 2-level multi-rate wavelet system, below left, is equivalent to the single-rate one, below right, in which all decimators are shifted past the filters:



Hence, from the right-hand diagram, the transfer function to the level-2 wavelet output (y_{01}) is $H_0(z)\ H_1(z^2)$ – i.e. this is the level-2 analysis basis function. Similarly, $G_1(z^2)\ G_0(z)$ is the level-2 reconstruction basis function. [25%]

(e) Smooth basis functions lead to reconstructed images with less visible artifacts, since sharp discontinuities are more visible than smooth features. By inspecting the two filter products in part (d), we see that $G_0(z)\ G_1(z^2)$ is smoother than $H_0(z)\ H_1(z^2)$, because the output of $G_0(z)\ G_1(z^2)$ linearly interpolates between the scaled values $\{-2,-4,12,-4,-2\}$ of the filter $G_1(z)$, whereas the output of $H_0(z)\ H_1(z^2)$ has some nasty *spikes* of value 14 and -8. For best coding performance we need the smooth filters to be in the reconstruction part of the coder, so it is best if the G filters stay as reconstruction filters. Therefore we should *not* swap the filters. [15%]

END OF SOLUTIONS

Version: NGK/3