Image Processing: Transforms, Filters and Applications

Dr. Anil C. Kokaram,

Electronic and Electrical Engineering Dept.,

Trinity College, Dublin 2, Ireland,

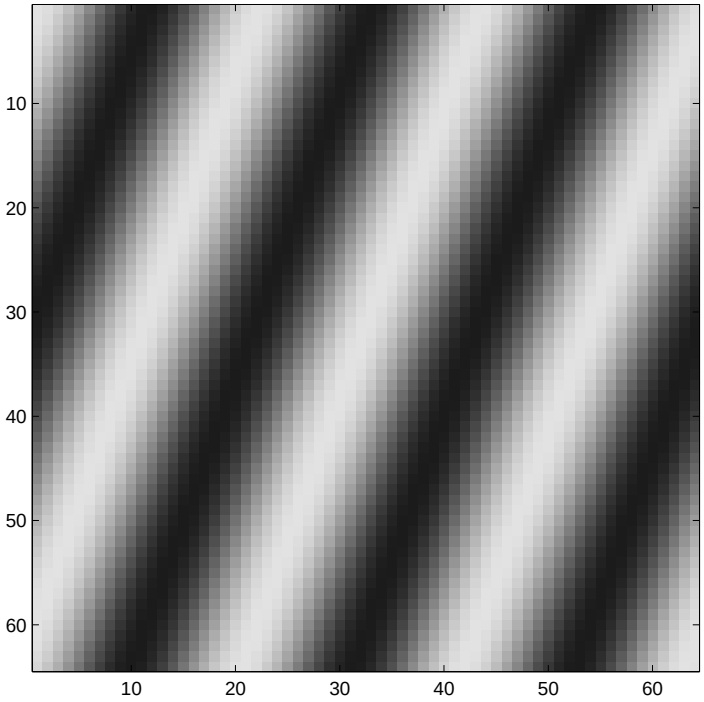
anil.kokaram@tcd.ie

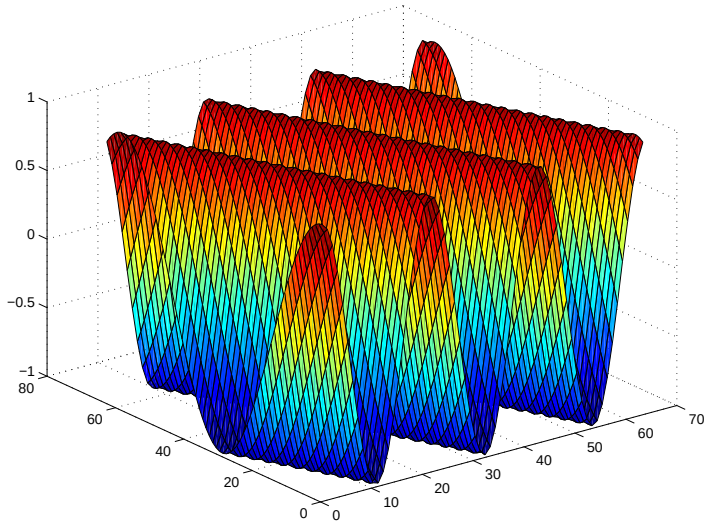
Transforms Overview

- 2D Fourier Transform
- Sampling Theorem
- 2D Z-transform
- 2D Filters and stability
- Some applications and the need for non-linear filters

2D Fourier Analysis

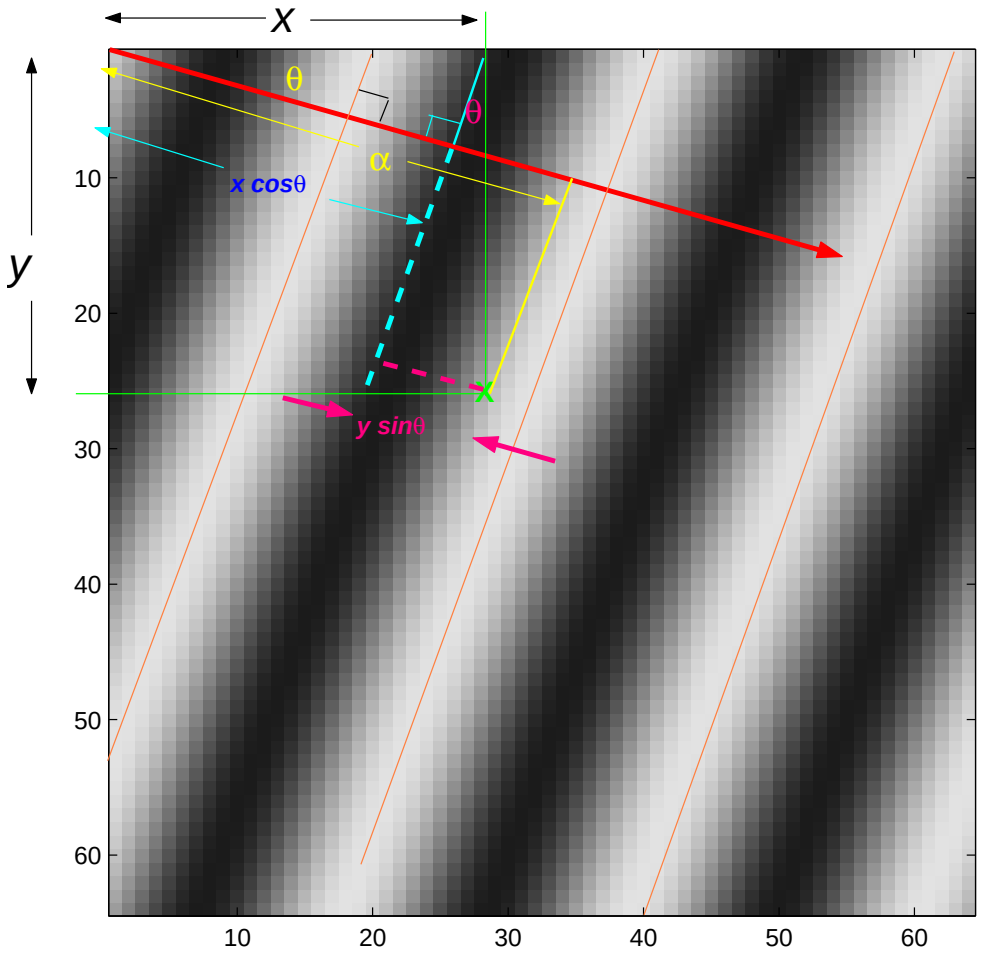
- Idea is to represent a signal as a sum of pure sinusoids of different amplitudes and frequencies.
- In 1D the sinusoids are defined by frequency and amplitude.
- In 2D these sinusoids have a *direction* as well e.g. $f(x, y) = a\cos(\omega_1 x + \omega_2 y + \phi)$





$a = 1.0$ $\omega_1 = 0.29$ $\omega_2 = 0.11$ Wave is directed at 20 degrees off horizontal, frequency is 0.05 cycles per pel *in that direction* and phase lag $phi = 0$.

How do ω_1, ω_2 relate to direction?



- In the direction θ: $f(\alpha) = a\cos(\omega_0\alpha + \phi)$, where ϕ is just some phase lag.
- Given any point (x, y), $\alpha = x\cos\theta + y\sin\theta$ Therefore:

$$f(x, y) = a\cos(\omega_0[x\cos\theta + y\sin\theta] + \phi)$$

Compare this with

$$f(x, y) = a\cos(\omega_1 x + \omega_2 y + \phi) \quad (1)$$

- Therefore $\omega_1 = \omega_0\cos\theta$ and $\omega_2 = \omega_0\sin\theta$
- Here $a = 1.0$; $\theta = 20^\circ$; $\phi = 45^\circ$; $\omega_0 = 0.05$cycles per pel.

The 2D Fourier Transform

- Recall 1D Fourier Transform of signal $x(t)$

$$X(\omega) = \int x(t) e^{-j\omega t} dt \tag{2}$$

- 2D Fourier Transform is similar. except with TWO frequency axes for horizontal and vertical frequency.

$$F(\omega_1, \omega_2) = \int_{y=-\infty}^{\infty} \int_{x=-\infty}^{\infty} f(x,y) e^{-j(\omega_1 x + \omega_2 y)} dx dy$$

$$f(x,y) = \frac{1}{4\pi^2} \int_{\omega_2=-\infty}^{\infty} \int_{\omega_1=-\infty}^{\infty} F(\omega_1, \omega_2) e^{j(\omega_1 x + \omega_2 y)} d\omega_1 d\omega_2 \tag{3}$$

Convolution: $f_1(x,y) \circledast f_2(x,y) \rightleftarrows F_1(\omega_1, \omega_2) F_2(\omega_1, \omega_2)$

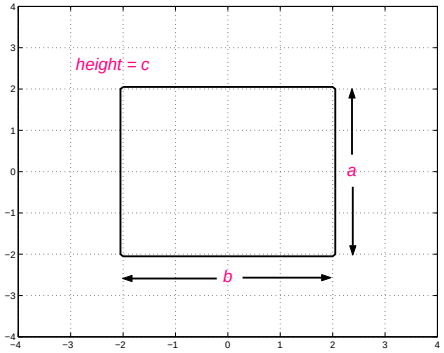
Parseval: $$\int_y \int_x |f(x,y)|^2 = \frac{1}{4\pi^2} \int_{\omega_2} \int_{\omega_1} F(\omega_1, \omega_2) F^*(\omega_1, \omega_2) d\omega_1 d\omega_2$$

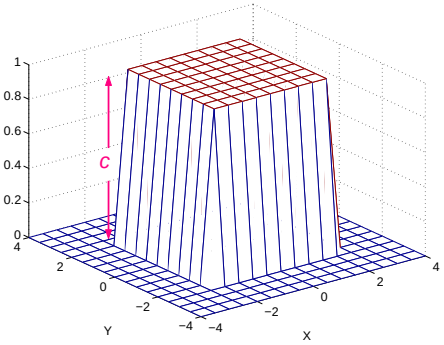
The 2D Fourier Transform is separable!

$$F(\omega_1,\omega_2) = \int_{y=-\infty}^{\infty}\int_{x=-\infty}^{\infty} f(x,y)e^{-j(\omega_1 x+\omega_2 y)}dxdy$$
$$= \int_{y=-\infty}^{\infty}\left[\int_{x=-\infty}^{\infty} f(x,y)e^{-j\omega_1 x}dx\right]e^{-j\omega_2 y}dy \quad (4)$$

- Can do 1D transform of rows first then do 1-D transform of result along columns. Or vice-versa.
- Fourier Xform of separable signals is also separable. Let $f(x,y) = f_x(x)f_y(y)$ then $F(\omega_1,\omega_2) = F_x(\omega_1)F_y(\omega_2)$. [Note that this is different from convolution identity because the signal is SEPARABLE!].

2D Fourier Xform example: $f(x,y) = \mathbf{rect}(a,b,c)$





$$F(\omega_1,\omega_2) = \int_y \int_x f(x,y) e^{-j(\omega_1 x + \omega_2 y)}\, dx \quad dy$$

$$= \int_{y=-a/2}^{a/2} \int_{x=-b/2}^{b/2} c e^{-j(\omega_1 x + \omega_2 y)}\, dx \quad dy$$

$$= \int_{y=-a/2}^{a/2} \left[\int_{x=-b/2}^{b/2} c e^{-j\omega_1 x}\, dx \right] e^{-j\omega_2 y}\, dy$$

$$\left[\int_{x=-b/2}^{b/2} c e^{-j\omega_1 x}\, dx \right] = c \left[\frac{e^{-j\omega_1 x}}{-j\omega_1} \right]_{-b/2}^{b/2}$$

$$= \frac{c}{-j\omega_1} \left[e^{-j\omega_1 b/2} - e^{j\omega_1 b/2} \right]$$

$$= 2c \frac{\sin(\omega_1 b/2)}{\omega_1} = bc \frac{\sin(\omega_1 b/2)}{\omega_1 b/2}$$

$$= bc\,\mathrm{sinc}\left(\frac{\omega_1 b}{2} \right)$$

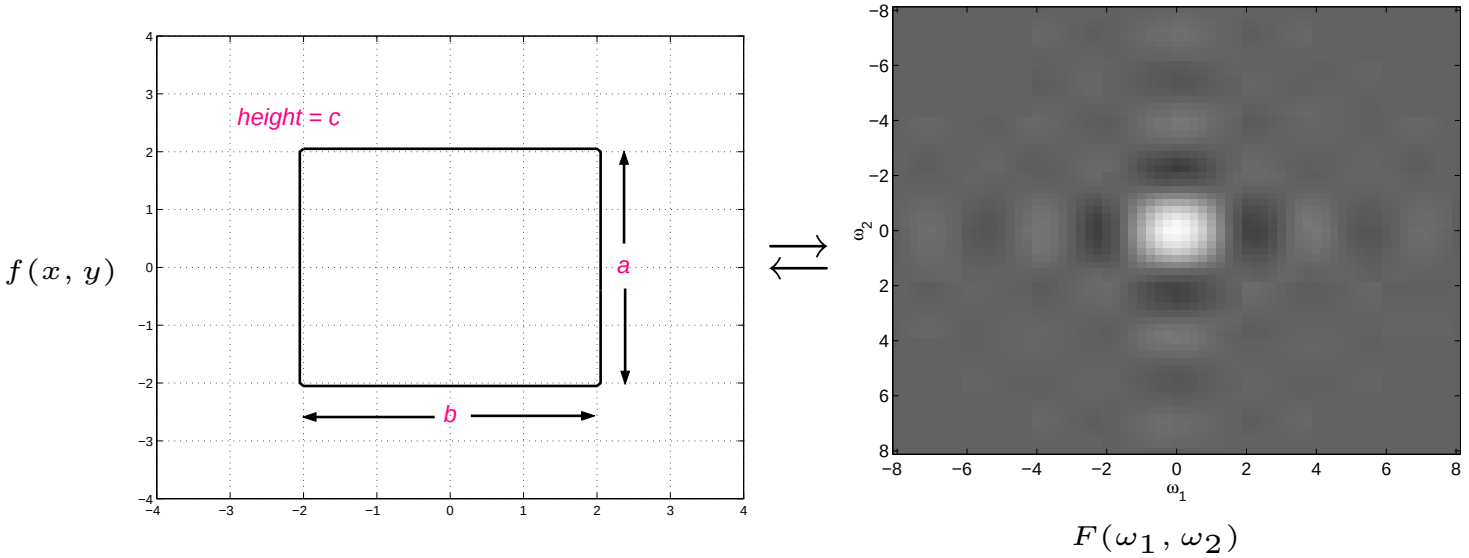
Vertical xform: $F(\omega_1,\omega_2) = \int_{-a/2}^{a/2} bc\,\mathrm{sinc}(\omega_1 b/2) e^{-j\omega_2 y}\, dy$

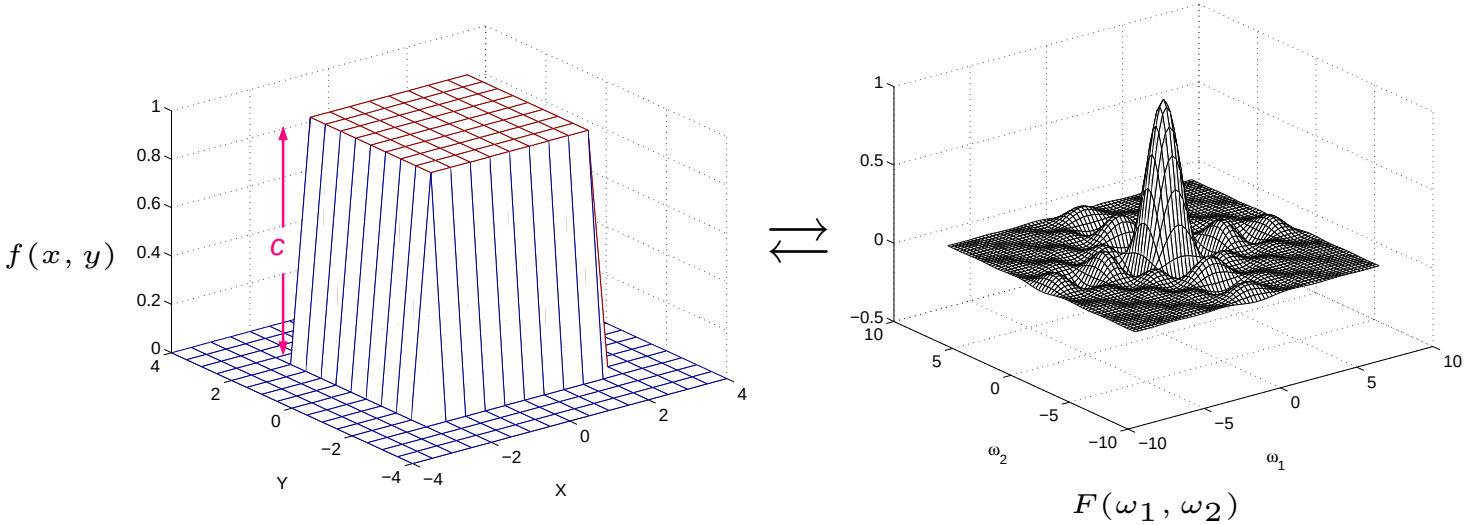
$$= bc\,\mathrm{sinc}(\omega_1 b/2) \int_{-a/2}^{a/2} e^{-j\omega_2 y}\, dy$$

$$= [bc\,\mathrm{sinc}(\omega_1 b/2)][a\,\mathrm{sinc}(\omega_2 a/2)]$$

$$= abc[\mathrm{sinc}(\omega_1 b/2)][\mathrm{sinc}(\omega_2 a/2)]$$

2D Fourier Xform example: $f(x,y) = \mathbf{rect}(a,b,c)$





More Fourier Identities (proofs ≈ similar to 1D)

$$\text{Given} \quad f(x,y) \rightleftarrows F(\omega_1,\omega_2)$$

$$\text{Shift Theorem} \quad f(x-d_x, y-d_y) \rightleftarrows F(\omega_1,\omega_2)e^{-j(\omega_1 d_x+\omega_2 d_y)}$$

$$\text{Coordinate Xformation} \quad f(\mathbf{Ax}) \rightleftarrows \frac{1}{||\mathbf{A}||}F(\left(\mathbf{A}^{-1}\right)^T \Omega)$$

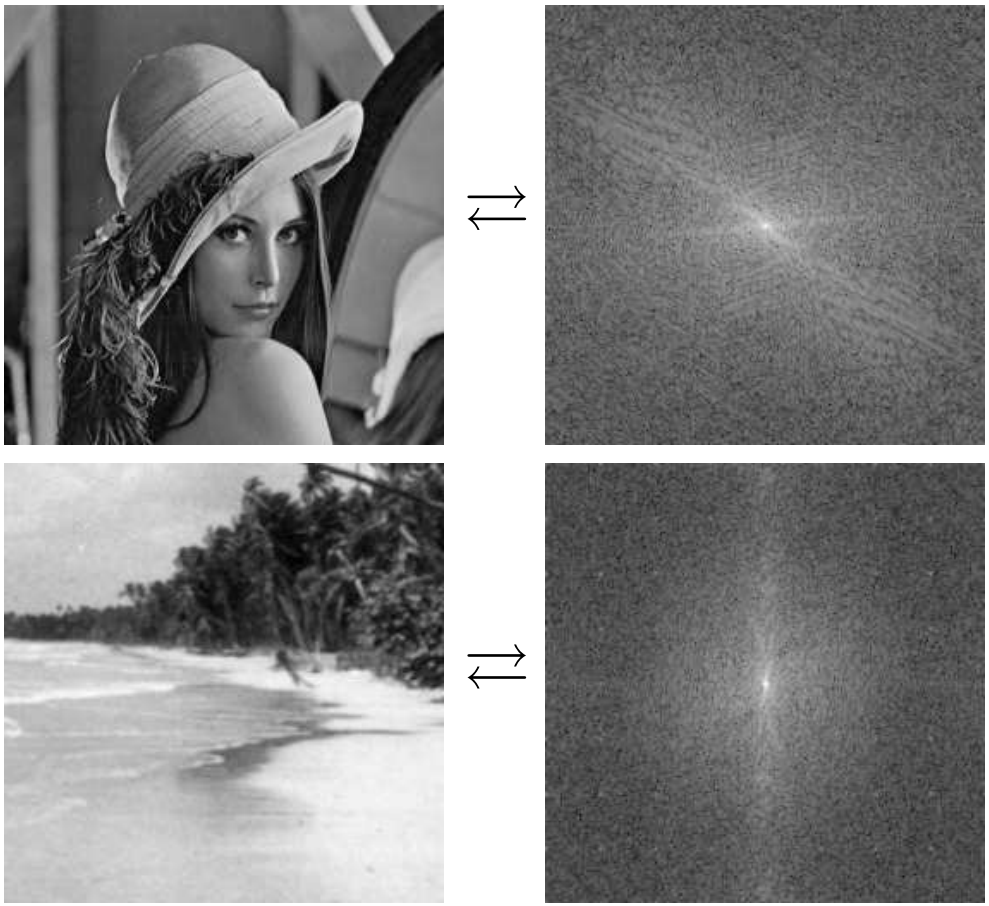
where $\mathbf{x} = [x \quad y]^T$ and $\mathbf{A}$ is a 2×2 matrix representing a general 2D transformation e.g.

$$\text{Zoom} \quad \mathbf{A} = \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix} \quad \text{Rotation} \quad \mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{5}$$

$$\text{Zoom} \quad (\mathbf{A}^{-1})^T = \begin{bmatrix} 1/a & 0 \\ 0 & 1/a \end{bmatrix} \quad \text{Rotation} \quad (\mathbf{A}^{-1})^T = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{6}$$

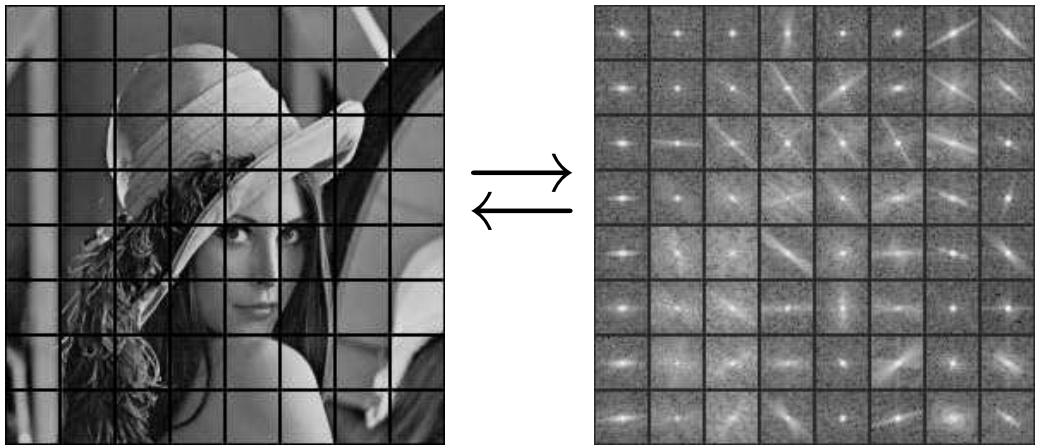
Therefore, magnifying the image by a factor a (zooming), causes the spectrum to *shrink* in area by that factor.
Rotating an image by θ causes the spectrum to rotate by the same angle.

Fourier Xform of images (Log Power Spectra dB)



- Most natural images have an exponential drop off in spectral power with increasing frequency
- Taking transform of signal causes *compaction* of energy into low-f regions

Fourier Xform of images (Log Power Spectra dB)



- Lena has been split into 64 32×32 blocks
- Images are not statistically homogenous over large areas
- Note how edges in the image correspond to highly directional spectra
- All the blocks have a strong dc component (i.e. average intensity is non-zero)
- This energy compaction is the key to the use of transforms for image compression
- Remember data windowing is an issue : used 2D Hamming here! (separable window)

Symmetry Property of FT and proof

For **REAL** signals:

$$F(\omega_1, \omega_2) = F^*(-\omega_1, -\omega_2)$$

PROOF Given $f(x,y) \rightleftarrows F(\omega_1, \omega_2)$

$$F(\omega_1, \omega_2) = \frac{1}{4\pi^2} \int_y \int_x f(x,y) e^{-j(\omega_1 x + \omega_2 y)} dx \quad dy \tag{7}$$

Take complex conjugate of both sides

$$F^*(\omega_1, \omega_2) = \frac{1}{4\pi^2} \int_y \int_x f^*(x,y) e^{j(\omega_1 x + \omega_2 y)} dx \quad dy$$

$f(x,y)$ is real $\Rightarrow$ $$= \frac{1}{4\pi^2} \int_y \int_x f(x,y) e^{j(\omega_1 x + \omega_2 y)} dx \quad dy$$

cf equation 7 $\Rightarrow$ $$= F(-\omega_1, -\omega_2) \quad \text{QED} \tag{8}$$

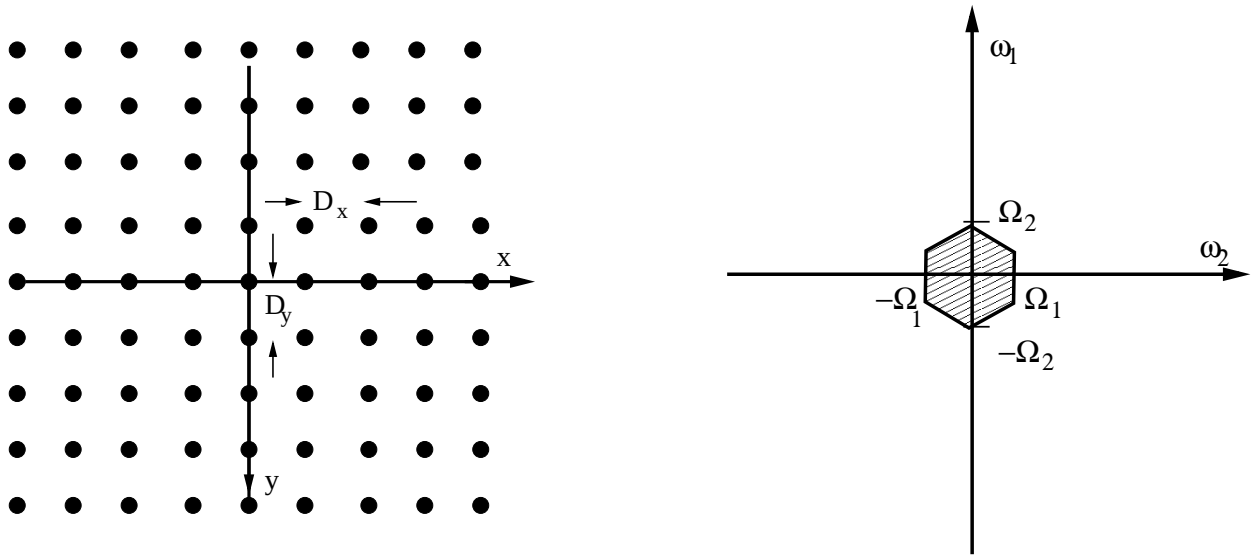
Just two quadrants specify the FT of an image with real valued pixels. See this from the pics previously

Sampling Theorem in 2D (similar arguments to 1D)

Consider a signal $f(x, y) \rightleftarrows F(\omega_1, \omega_2)$ sampled by a grid of delta functions $s(x, y)$

$$s(x, y) = D_x D_y \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \delta(x - n_1 D_x, y - n_2 D_y) \tag{9}$$

where D_x, D_y are the separations between points on the sampling grid i.e. the *period* of horizontal and vertical sampling, and $D_x D_y$ is just used as a normalising constant. Sampling (or digitization) is modelled as $f_s(x, y) = f(x, y)s(x, y)$. Now want to find out the relationship between $F(\omega_1, \omega_2)$ and $F_s(\omega_1, \omega_2)$.



Left: $s(x, y)$: each dot is a $\delta-$ function coming out of the plane of the page. Right: $F(\omega_1, \omega_2)$, we assume that the spectrum of $f(x, y)$ is bandlimited between $\pm\Omega_1$ and $\pm\Omega_2$ respectively.

Convenient to use 2D Fourier series to express $s(x,\,y)$ (since it's a periodic function) before proceeding.

$$s(x,\,y) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} a_{k_1,k_2}\, e^{j(k_1\omega_1^0 x + k_2\omega_2^0 y)} \tag{10}$$

where $\omega_1^0,\quad \omega_2^0 = \frac{2\pi}{D_x},\quad \frac{2\pi}{D_y}$ respectively i.e. the fundamental frequency of $s(x,\,y)$ in the two directions.

$$\begin{aligned} a_{k_1,k_2} &= \frac{1}{D_x D_y} \int_y \int_x s(x,\,y) e^{-j(k_1\omega_1^0 x + k_2\omega_2^0 y)}\, dx\; dy \\ &= \frac{1}{D_x D_y} \int_{y=-D_y/2}^{D_y/2} \int_{x=-D_x/2}^{D_x/2} D_x D_y\, \delta(x,\,y) e^{-j(k_1\omega_1^0 x + k_2\omega_2^0 y)}\, dx\; dy \\ &= \frac{1}{D_x D_y} D_x D_y = 1 \text{ By sifting property of } \delta- \text{ function} \end{aligned} \tag{11}$$

Therefore, substituting for a_{k_1,k_2} in equation 10 we can state that

$$s(x,\,y) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} e^{j(k_1\omega_1^0 x + k_2\omega_2^0 y)} \tag{12}$$

We can now proceed to find $F_s(\omega_1, \omega_2)$ as follows

$$F_s(\omega_1, \omega_2) = \int_y \int_x f_s(x, y) e^{-j(\omega_1 x + \omega_2 y)} dx \quad dy$$

$$= \int_y \int_x s(x, y) f(x, y) e^{-j(\omega_1 x + \omega_2 y)} dx \quad dy$$

$$\text{From eq. 12} = \int_y \int_x \left[\sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} e^{j(k_1 \omega_1^0 x + k_2 \omega_2^0 y)} \right] f(x, y) e^{-j(\omega_1 x + \omega_2 y)} dx \quad dy$$

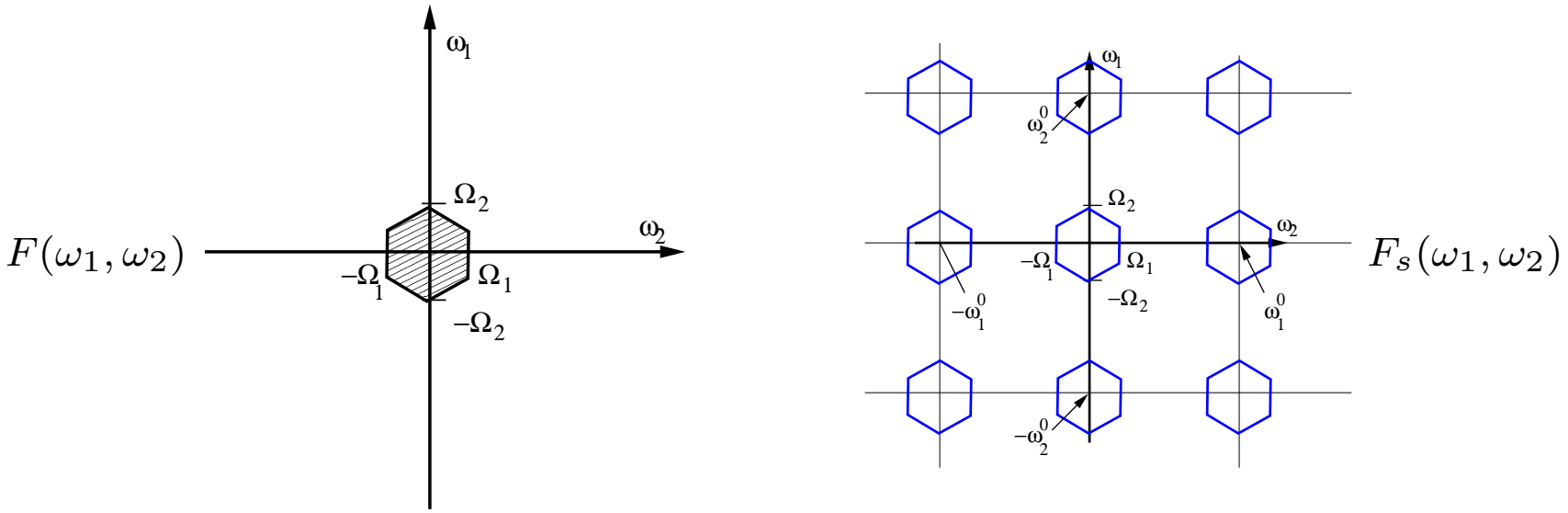
$$\text{(Can swap summations)} = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \int_y \int_x f(x, y) e^{-j(\omega_1 x + \omega_2 y)} e^{j(k_1 \omega_1^0 x + k_2 \omega_2^0 y)} dx \quad dy$$

$$\text{(Collect common terms)} = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \int_y \int_x f(x, y) e^{-j([\omega_1 - k_1 \omega_1^0] x + [\omega_2 - k_2 \omega_2^0] y)} dx \quad dy$$

Integral Same form as if ω_1, ω_2 were replaced with $(\omega_1 - k_1 \omega_1^0)$ etc

$$\text{SO: } F_s(\omega_1, \omega_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} F(\omega_1 - k_1 \omega_1^0, \omega_2 - k_2 \omega_2^0)$$

2D Sampling Theorem



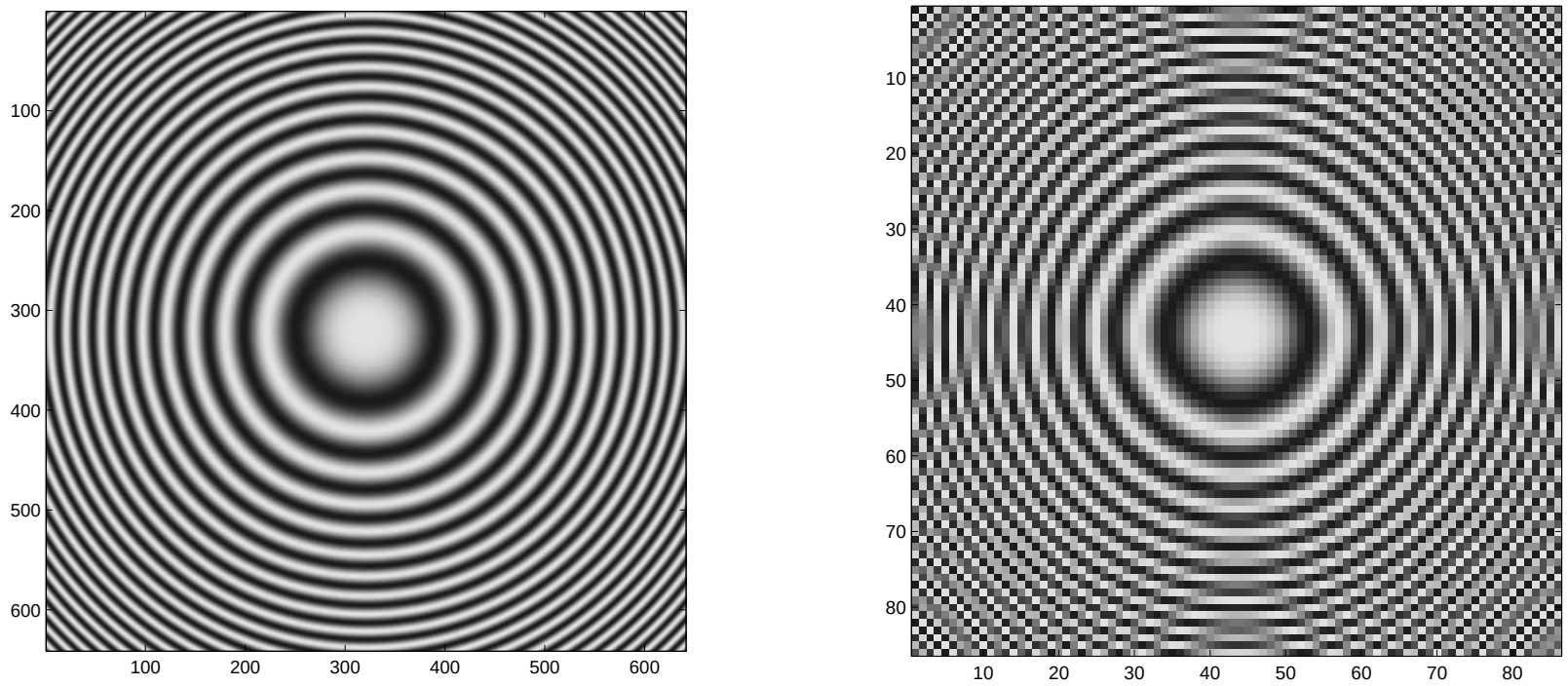
For no overlap in the spectra to occur, $\omega_1^0 - \Omega_1 > \Omega_1$ and $\omega_2^0 - \Omega_2 > \Omega_2$. Thus we get Nyquist's theorem for 2D (its similar to 1D).

$$\omega_1^0 > 2\Omega_1 \quad \text{AND} \quad \omega_2^0 > 2\Omega_2 \qquad (13)$$

for no aliasing to occur.

But the sampling frequency of any picture will vary with the distance of the observer from the display (remember cycles/degree on eye?). So pictures which are not aliased when viewed from far away can become aliased when viewed at closer distances! [Note: this is a crude simplification of the effects involved.]

Aliasing



Zonneplate image $s(x, y) = \cos(a_x x^2 + a_y y^2)$ where $a_x = b_x = 0.0628$

LEFT: Zoneplate sampled at 5× rate on RIGHT. LEFT image is ok (smooth concentric circles seen), RIGHT image shows aliasing: ghost shapes, pixellation. Effect is worse at the higher frequencies near the border of the image.

Aliased Lena

Left hand image sampled at 4× higher rate than right.

An Aliasing problem in archive TV and film

Resampling of TV footage recorded on film gives aliased pictures.

2D Discrete Fourier Xform

- As expected, the 2D DFT is similar to the 1D version and is a **separable** transform

Given an image of size $M \times N$ pixels, the 2D DFT is defined as

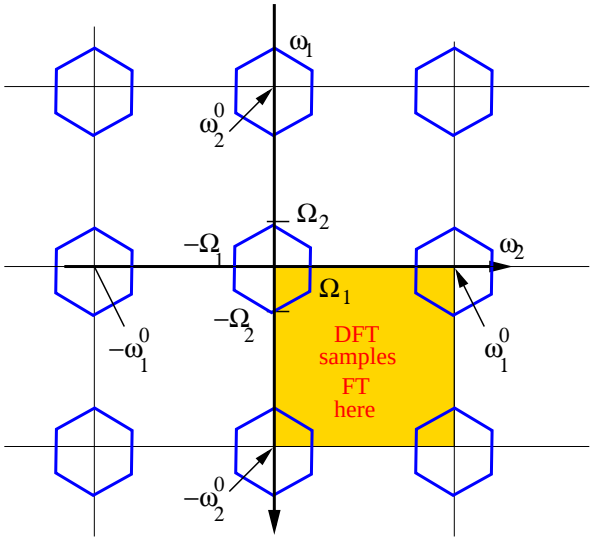
$$F(h,k) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f(n,m) e^{-j\left(\frac{2\pi}{N} nh + \frac{2\pi}{N} mk\right)}$$

$$f(n,m) = \frac{1}{NM} \sum_{h=0}^{N-1} \sum_{k=0}^{M-1} F(h,k) e^{j\left(\frac{2\pi}{N} nh + \frac{2\pi}{N} mk\right)} \qquad (14)$$

For $0 \leq h \leq N-1$, $0 \leq k \leq M-1$ and $0 \leq n \leq N-1$, $0 \leq m \leq M-1$

- Fast Algorithm exists (Lim pages 163–182). Tedious to derive, won't bother, read the book if you have to make devices/implementation. Note that most DSP manufacturers supply FFT libraries, TI, Analog, Intel, (Matlab : `fft, fft2`) all do. FFTW (Fastest Fourier Transform in the West) developed at MIT good for general purpose processors see FFTW web site.

Symmetry Properties and other practical stuff



- DFT calculates spec. components at equal intervals up till the sampling frequency. Given $N \times M$ transform: $(0,0)th$ bin is dc coeff and $N-1$th bin $\equiv 2\pi/D_y$ similarly for $M-1$th bin.
- The FFT algorithms are optimal for power of 2 data lengths and typically return the DC coeff in the first bin. Can zero pad to get more frequency bins if needed e.g. for fast convolution [but note no extra information acquired].
- Need to shuffle coefficients to get zero freq at centre. See what `fftshift, fftshift2` do.
- Symmetry about DC: $F(h,k) = F((N-h) \text{ MOD } N, (M-k) \text{ MOD } M)$.
- Can express transform as a matrix operation on rows then columns

$$\mathcal{I}_{\mathcal{F}} = \mathbf{W}^T \mathbf{I} \mathbf{W} \tag{15}$$

The 2D z-transform

- Recall 1D Z Transform of signal x_n

$$X(z) = \sum_{n=-\infty}^{\infty} x_n z^{-n} \tag{16}$$

  Just a polynomial in z (a complex number). Used to solve difference equations. Also helps with stability of IIR filters.

- Z-transform of a sequence $g(h,k)$ is denoted $G(z_1, z_2)$.

$$G(z_1, z_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} g(n_1, n_2) z_1^{-n_1} z_2^{-n_2} \tag{17}$$

  The z_1 and z_2 components act on the sequence $p(h,k)$ along vertical and horizontal directions in a sense.

- Z transform in 2D is a function of 2 complex numbers z_1, z_2. It exists in a 4D space so is very difficult to visualise.

- Z-transform of a sequence is related to the spectral content of the sequence as follows:

$$G(\omega_1, \omega_2) = G(e^{j\omega_1}, e^{j\omega_2}) \tag{18}$$

- Given a sequence $g(h,k) =$ `[1 2;3 4]`. Top left is [0,0];

$$G(z_1, z_2) = 1 + 2z_2^{-1} + 3z_1^{-1} + 4z_1^{-1}z_2^{-1} \tag{19}$$

- Given signal $X(z_1, z_2)$ input into a system with transfer function $H(z_1, z_2)$ output $Y(z_1, z_2)$ is $X(z_1, z_2)H(z_1, z_2)$ [Just like 1D]
- $H(z_1, z_2) = 4 - z_1^{-1} - z_1 - z_2^{-1} - z_2$ is an FIR filter.
- See this from difference equation if you like: $Y(z_1, z_2) = X(z_1, z_2)H(z_1, z_2)$ therefore:

$$\begin{aligned} Y(z_1, z_2) &= X(z_1, z_2)(4 - z_1^{-1} - z_1 - z_2^{-1} - z_2) \\ &= 4X(z_1, z_2) - z_1^{-1}X(z_1, z_2) - z_1X(z_1, z_2) - z_2^{-1}X(z_1, z_2) - z_2X(z_1, z_2) \\ \Rightarrow \quad y(h,k) &= 4x(h,k) - x(h-1,k) - x(h+1,k) - x(h,k-1) - x(h,k+1) \end{aligned}$$

  $y(h,k)$ only depends on input $x(h,k)$, so when input stops, output stops.

- Or work out impulse response $P(z_1, z_2) \rightleftarrows p(h,k)$. Input is then

$x(h,k) = [1\ 0\ 0\ \ldots; 0\ 0\ 0\ \ldots;\ 0\ 0\ \ldots; \ldots]$ So $X(z_1, z_2) = 1$ and then

$$P(z_1, z_2) = X(z_1, z_2)H(z_1, z_2) = 4 - z_1^{-1} - z_1 - z_2^{-1} - z_2$$

$$\Rightarrow \quad p(h,k) = \mathcal{Z}^{-1}\Big(P(z_1, z_2)\Big) \equiv \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{20}$$

- Note that impulse response of a filter *is its convolution mask*!.

IIR filter

$$H(z_1, z_2) = \frac{H_1(z_1, z_2)}{H_2(z_1, z_2)} \qquad \text{For example:} \quad H(z_1, z_2) = \frac{1}{4 - z_1^{-1} - z_1 - z_2^{-1} - z_2} \tag{21}$$

Corresponding difference equation given input $x(h,k)$ and output defined as $y(h,k)$:

$$\begin{aligned} Y(z_1,z_2) &= X(\cdot)H(\cdot) \\ &= \frac{X(z_1,z_2)}{(4-z_1^{-1}-z_1-z_2^{-1}-z_2)} \\ \Rightarrow \quad Y(z_1,z_2)(4-z_1^{-1}-z_1-z_2^{-1}-z_2) &= X(z_1,z_2) \\ \Rightarrow \quad y(h,k) &= \end{aligned}$$

$$\frac{1}{4}\left(x(h,k)+y(h-1,k)+y(h+1,k)\ +\ y(h,k-1)+y(h,k+1)\right) \qquad (22)$$

Now output depends on previous outputs *and* outputs to come! IIR filter.

Stability of filters

LIM pages 102–123

Stability of LSI systems requires that the impulse response $p(h,k)$ of the system is finitely summable i.e.

$$\sum_{h=-\infty}^{h=\infty} \sum_{k=-\infty}^{k=\infty} |p(h,k)| < \infty \tag{23}$$

- In 1D this criterion is exposed by examining the poles of the system and their position relative to the unit circle.
- Stability of non-separable 2D IIR filters is difficult to analyse. There is no polynomial factorisation theorem for n-D, and we have a 4-D Z-Xform surface! No notion of poles because $H_2(z_1, z_2) = 0$ maps out a SURFACE.
- We shall not bother with non-separable 2D IIR filter stability for this reason.

Separable filters and stability

We already dealt with filters that had a separable impulse response. Their system xfer function is then

$$H(z_1, z_2) = H_1(z_1)H_2(z_2) \tag{24}$$

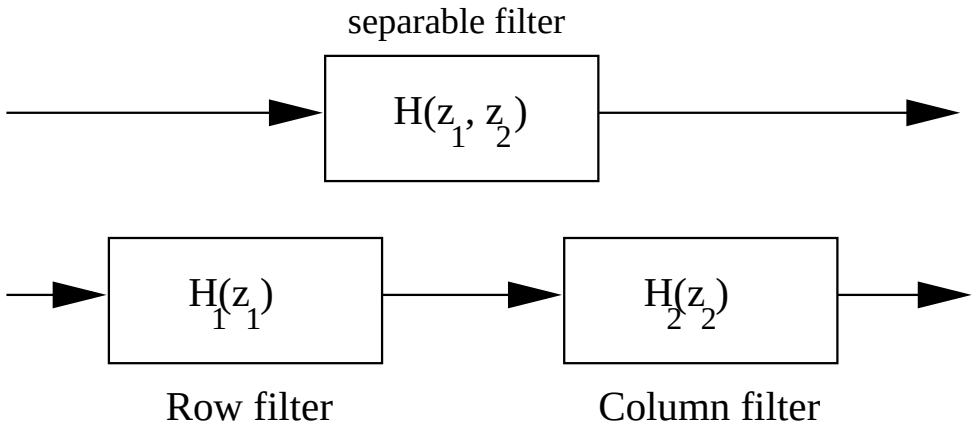
These types of systems are implemented by processing along rows (or columns) and then processing along columns (or rows). Revisit that low pass filter again.

Given $H_1(z_1) = z_1^{-1} + 2 + z_1$ and $H_2(z_1) = z_2^{-1} + 2 + z_2$. What is the impulse response of the system $H_1(z_1, z_2)H_2(z_1, z_2)$?

$$\begin{aligned} H(z_1, z_2) &= H_1(z_1)H_2(z_2) \\ &= (z_1^{-1} + 2 + z_1)(z_2^{-1} + 2 + z_2) \\ &= z_1^{-1}z_2^{-1} + 2z_1^{-1} + z_1^{-1}z_2 + 2z_2^{-1} + 4 + 2z_2 + z_1z_2^{-1} + 2z_1 + z_1z_2 \end{aligned}$$

$$\Rightarrow \quad \mathcal{Z}^{-1}\left(H(z_1, z_2)\right) \equiv \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{25}$$

Separable filters and stability



- Stability analysis of separable filters is therefore conducted in the same manner as 1-D filters
- Just have to make sure that the row and column filters are separately STABLE, and then the whole system is stable. Can use usual pole/zero analysis in these cases.

Filter Design: Importance of Zero Phase

Zero Phase Low Pass Filter

Non-zero Phase Low Pass Filter

- Typically usable filters for images have a Zero (or linear) phase response
- This implies that $H(\omega_1, \omega_2) = H^*(\omega_1, \omega_2)$ i.e. the frequency response is a REAL function. And as with 1D filters this implies that $p(h,k) = p*(-h,-k)$. This means that the magnitude of the filter impulse response is also *symmetric*. For real filters this condition implies that $p(h,k) = p(-h,-k)$.

Filter Design

- Can use 1D methods for separable filters.
- FIR filter design similar to 1D in general i.e. Can design in frequency domain, take inverse then window the impulse response; or can specify problem as constrained optimisation
- In practice can only afford short filters if seeking real time performance. This limits the effort that is sensible for filter design.
- Longer filters tend to run into problems due to the non-stationary statistics of images. Typically, want to use a linear filter to achieve some effect, but that effect should vary across the image.
- 2D Filter design gained in importance with wavelet analysis
- Will look at one simple example only: Designing a low pass filter to prevent aliasing when downsizing an image.

Application: Anti-Aliasing filter for factor of $\mathcal{N}$ downsampling

Recall (from sampling discussion) that initial sampling rate is D_x, D_y pels per metre for original signal $f(h,k)$. Given sampling rate is the same in both directions $D_x = D_y = D$. So sampling frequency is $[\Omega_0,\ \Omega_0] = [(2\pi/D),\ (2\pi/D)]$. Want to reduce this to $[\Omega_0/\mathcal{N},\ \Omega_0/\mathcal{N}]$ for factor of $\mathcal{N}$ downsampling in both directions to give new signal $f_{\mathcal{N}}(h,k)$.
Spectrum of sampled signal contains copies of original spectrum $f(\omega_1,\omega_2)$ at integer multiples of $[\Omega_0,\ \Omega_0]$. If sampling rate is reduced to $[\Omega_0/\mathcal{N},\ \Omega_0/\mathcal{N}]$ then need to ensure that there are no spectral components above $[\Omega_0/(2\mathcal{N}),\ \Omega_0/(2\mathcal{N})]$.
So need to design a low pass filter $P(z_1,z_2) \rightleftarrows p(h,k)$ with cut-off at $[\Omega_0/(2\mathcal{N}),\ \Omega_0/(2\mathcal{N})]$. Use this to low pass $f(h,k)$ then subsample by factor $\mathcal{N}$ in both directions to produce $f_{\mathcal{N}}(h,k)$. Use window method. Use perfect low pass filter $\text{rect}(\Omega,\Omega,1) = \text{rect}(\Omega_0/(2\mathcal{N}),\ \Omega_0/(2\mathcal{N}),1)$. Find inverse FT to give (continuous) impulse response of filter, then window with a 2D Hamming or Hanning or Kaiser or ... window. We'll use Hamming

$$
\begin{aligned}
p(x,y) &= \frac{1}{4\pi^2}\int\int \text{rect}(\Omega,\Omega,1)e^{j(\omega_1 x+\omega_2 y)}\,dx\quad dy \\
&= \frac{1}{4\pi^2}\int_{-\Omega}^{\Omega}\int_{-\Omega}^{\Omega} e^{j(\omega_1 x+\omega_2 y)}\,dx\quad dy \\
&= \frac{1}{4\pi^2}\left(\frac{2}{x}\sin(x\Omega)\right)\int_{-\Omega}^{\Omega} e^{j(\omega_1 y)}\,dy \\
&= \frac{1}{4\pi^2}\frac{2}{x}\sin(x\Omega)\frac{2}{y}\sin(y\Omega) \\
&= \frac{1}{4\pi^2}2\Omega\text{sinc}(x\Omega)2\Omega\text{sinc}(y\Omega) \qquad (26)
\end{aligned}
$$

Filter is separable !

Application: Anti-Aliasing filter for factor of $\mathcal{N}$ downsampling

$$p(x,y) = \frac{1}{4\pi^2} 2\Omega \text{sinc}(x\Omega) 2\Omega \text{sinc}(y\Omega)$$

But need to apply filter in digital domain. Recall sampling impulse train $s(x,y)$ previously. Samples at $y = hD \quad x = kD$ and recall scaling factor $1/(D \times D)$. Also $\Omega = \Omega_0/(2\mathcal{N}) = (2\pi/(2D\mathcal{N}) = (\pi/(D\mathcal{N})$

$$\begin{aligned} p(h,k) &= \frac{1}{4\pi^2} \left[\frac{2\pi}{D\mathcal{N}} \text{sinc}\left(\frac{hD\pi}{D\mathcal{N}} \right) \right] \left[\frac{2\pi}{D\mathcal{N}} \text{sinc}\left(\frac{kD\pi}{D\mathcal{N}} \right) \right] \\ &= \left[\frac{1}{D\mathcal{N}} \text{sinc}\left(\frac{h\pi}{\mathcal{N}} \right) \right] \left[\frac{1}{D\mathcal{N}} \text{sinc}\left(\frac{k\pi}{\mathcal{N}} \right) \right] \end{aligned}$$

Say that output signal is $g(h,k) = f(h,k) \circledast p(h,k)$, then

$$g(h,k) = f(h,k) \circledast p(h,k) = D \times D \sum_h \sum_k f(y - hD, x - kD) \circledast p(h,k)$$

$$= \sum_h \sum_k f(y - hD, x - kD) \circledast D \times Dp(h,k)$$

$$\text{Thus} \quad p(h,k) = \left[\frac{1}{\mathcal{N}} \text{sinc}\left(\frac{h\pi}{\mathcal{N}}\right)\right]\left[\frac{1}{\mathcal{N}} \text{sinc}\left(\frac{k\pi}{\mathcal{N}}\right)\right]$$

So filter in digital domain is scaled by sampling frequency and so does not depend on the actual value of the sampling frequency. (No different from 1D filters)

But sinc function is infinitely long so need to multiply impulse response by some data window $w(h,k)$, to reduce filter length intelligently to length N say. Use separable window $w(h)w(k)$. Thus

$$p(h,k) = \left[\frac{w(h)}{\mathcal{N}} \text{sinc}\left(\frac{h\pi}{\mathcal{N}}\right)\right]\left[\frac{w(k)}{\mathcal{N}} \text{sinc}\left(\frac{k\pi}{\mathcal{N}}\right)\right]$$

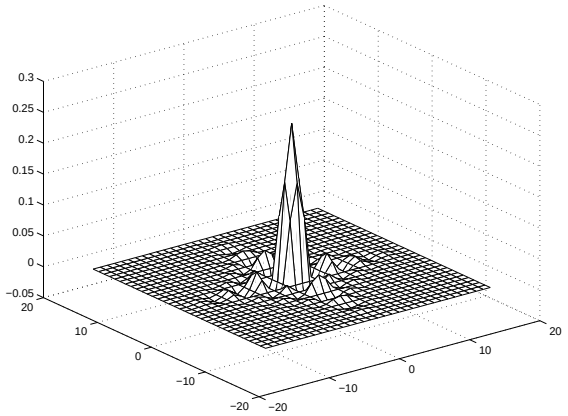
where

$$w(m) = \begin{cases} 0.54 + 0.46\cos(\frac{\pi m}{N}) & |m| < N \\ 0 & \text{Otherwise} \end{cases} \tag{27}$$

See Lim Chapter 4 for FIR Filter Design and better design method: frequency xformation pages 218–238.

Implementation

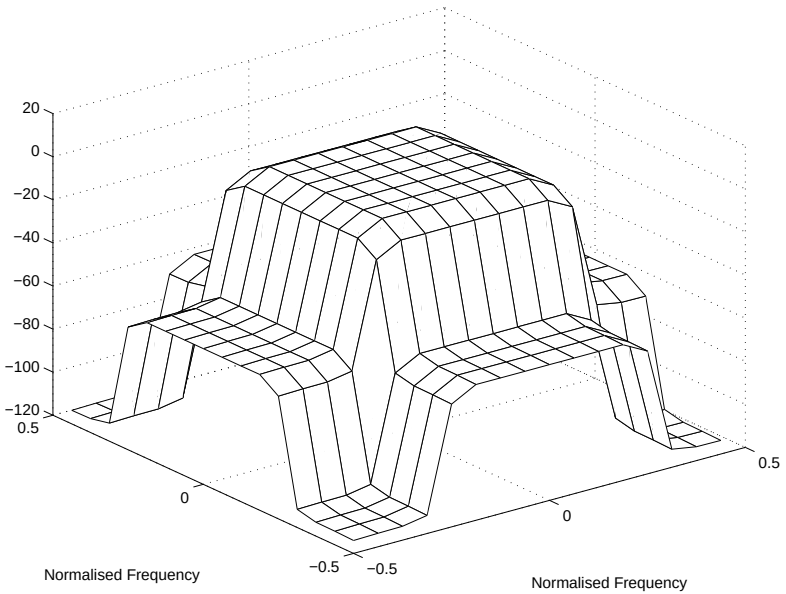
$p(h,k)$ 33 $\times$ 33 tap filter



FOR the separable filters:

```
N = 2; % subsampling factor
n = (-16:16);
p = (1/(N))*sinc(n/(N));
% set window
w = hamming(length(n))';
% window the impulse response to get filter p
p = p.*w;
p = p/sum(sum(p)); % Make sure that DC gain of p
is 1.0
```

$P(\omega_1, \omega_2)$



Implementation

Original

A) Subsampled $\mathcal{N} = 2$ with anti aliasing

B) Subsampled $\mathcal{N} = 2$ without anti aliasing

B

A

Is this really necessary?

- SINC filter has to be quite long to do a reasonable job. (Used 33 taps in previous example). Can cause ringing at edges, bad for image perception.
- Can use simple average of 5×5 pel window, or Gaussian shaped filter (also separable). Results acceptable for many apps.

$$p(h,k) \propto \exp - \left(\frac{h^2}{2\sigma_h^2} \right) \exp - \left(\frac{k^2}{2\sigma_k^2} \right) \tag{28}$$

  where σ_h, σ_k affect the filter bandwidth. Bigger σ implies bigger bandwidth. Need to normalise the filter coeffs so that they sum to 1.0 for low pass effect. `p = p/sum(sum(p))`.
- BTW Good zooming or upsampling (needed for deinterlacing or TV→Cinema conversion) is *much* harder. A simple example is used for your first assignment.

Application: Noise Reduction (Low pass vs. Wiener)

Given an observed signal $g(h,k) = f(h,k) + \eta(h,k)$ where $f(h,k)$ is the clean original and $\eta(h,k) \sim \mathcal{N}(0, \sigma_\eta^2)$; how do we recover $f(h,k)$ from $g(h,k)$?

- Can take approach that noise is visible because it creates a significant high frequency component in the otherwise 'smooth' image. So just using a low pass filter to reduce the high-freq content of $g(h,k)$ should do the trick.
- But that kills image features like edges and so causes a reduction in sharpness.
- Can use Frequency Domain Wiener filter $H(\omega_1, \omega_2)$ (pages 354–356 Lim) i.e. $\hat{F}(\omega_1, \omega_2) = G(\omega_1, \omega_2)H(\omega_1, \omega_2)$. The filter can be defined in terms of the PSD of the signal $f(\cdot)$ and noise $e(\cdot)$ as

$$H(\omega_1,\omega_2) = \frac{P_{gf}(\omega_1,\omega_2)}{P_{gg}(\omega_1,\omega_2)} = \frac{P_{ff}(\omega_1,\omega_2)}{P_{ff}(\omega_1,\omega_2) + P_{\eta\eta}(\omega_1,\omega_2)}$$
$$= \frac{P_{gg}(\omega_1,\omega_2) - P_{\eta\eta}(\omega_1,\omega_2)}{P_{gg}(\omega_1,\omega_2)} = 1 - \frac{P_{gg}(\omega_1,\omega_2)}{P_{\eta\eta}(\omega_1,\omega_2)} \quad (29)$$

- Since η is white Gaussian noise, its PSD is a constant across the whole spectrum. This constant value $\propto \sigma_\eta^2$.

- Can implement Wiener filter using $|\text{DFT}|^2$ to estimate Power Spectrum of $g(\cdot)$.
- BUT to estimate DFT of $g(\cdot)$ need to window the data with some analysis window to prevent spectral leakage. Use 2D Hamming say (separable).

Application: Noise Reduction (Low pass vs. Wiener)

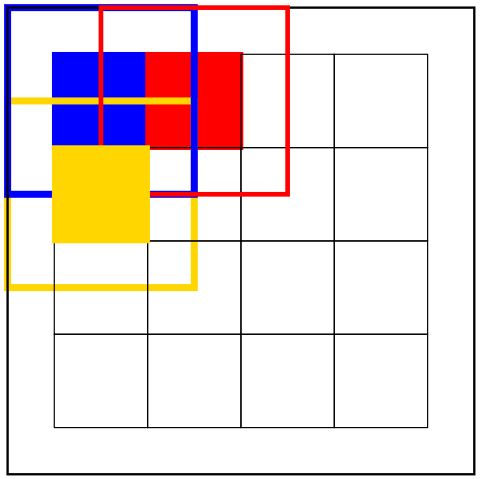
Noisy Lena $\sigma^2_{\eta\eta} = 100$ about 28dB SNR

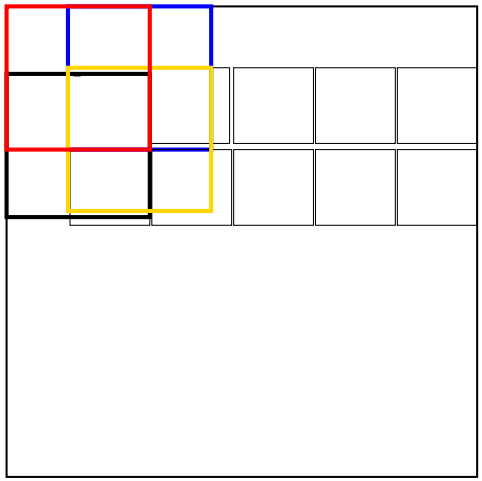
Wiener result, Hamming analysis window using $\sigma^2_{\eta\eta} = 500$

Gaussian filter, 11×11 tap $\sigma^2_h = \sigma^2_k = 1.5$

Implementation: Block based image processing

- Problem is that underlying image is statistically non-stationary. Can approximate as stationary in blocks. So process blocks independently then either tile the output or overlap the output. Can get better edge preservation this way.





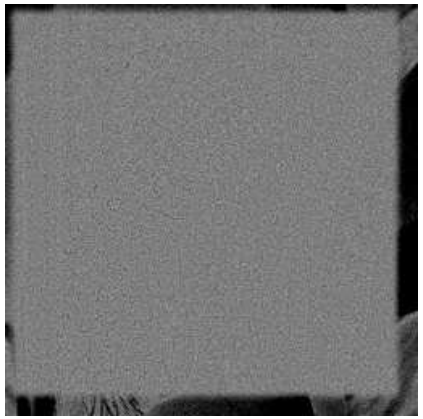
Left: Overlapped Tiling (overlap/save). Right: Overlap/add (2:1 overlap)

Implementation: Block based image processing

Noisy Lena $\sigma^2_{\eta\eta} = 100$ about 28dB SNR

Wiener result with one big block

Wiener filter with overlapped processing



Difference with noisy original

Application: Impulse noise reduction using a non-linear filter

- Impulsive noise cannot be easily removed with linear filters. Distortion occurs only at a subset of pel sites and tends to take form of data dropout or data obliteration.
- Median filter very useful in image processing for these kinds of defects. Idea is to define some $N \times N$ filter geometry and then the output of the filter is the median of the pixels in its window.



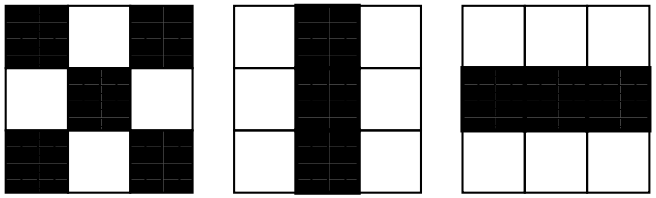
Noisy, Filtered with 5×5 median filter, Filtered with 5×5 average filter

Application: Impulse noise reduction using a non-linear filter

- Problem is that median filter preserves gross image edges but ejects fine texture. Causes a kind of flattening of detail.
- This filter is one from the class of *Order statistic* filters. The output of this general class is some kind of statistic measure of the pels in its window e.g. Upper Quartile, Lower Quartile etc. Need to calculate histogram of pels in the window then select the pel at Q_1 or Q_3 etc.
- Around 1990 a few researchers (Arce et al) spotted that you could improve the performance of the median operator by *cascading* filter geometries. These are called **multistage** median filters.

1	10	3
200	14	14
15	14	14

Median filter output = 14



Summary

This has been a big section.

- Looked at Fourier Analysis in 2D
- Noticed that z-transform for 2D exists but only really useful for examining FIR filter stability
- Looked at a simple filter design example
- Fast overview of some application and practical matters regarding linear filters for images including linear phase
- Median filter was introduced as an example of a non-linear image filter