Image Processing: Transforms (part 2)

4C8: Digital Media Processing

Ussher Assistant Professor François Pitié

2016/2017

Department of Electronic & Electrical Engineering , Trinity College Dublin

adapted from original material written by Prof. Anil Kokaram.

Transforms Overview

- Sampling Theorem
- Downsampling
- Filter Design & Examples
- The need for non-linear filters

Additional reference for this handout can be found in the book *Two-Dimensional Signal and Image Processing by Jae. S. Lim and* Applied Digital Signal Processing by Manolakis and Ingle.

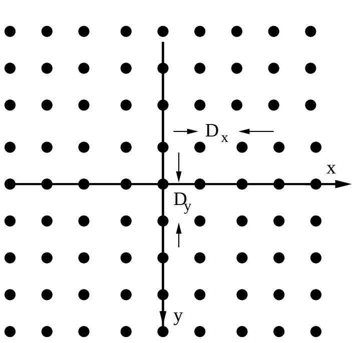
Sampling Theorem

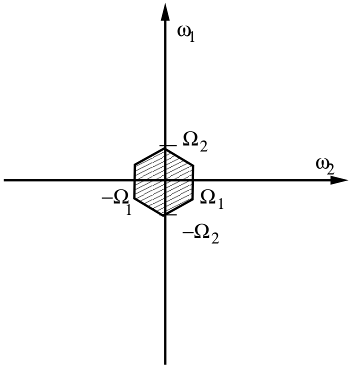
Sampling Theorem in 2D (similar arguments to 1D)

Consider a signal $f(x,y) \rightleftarrows F(\omega_1,\omega_2)$, we wish to represent $f(x,y)$ using a discrete sequence $f[m,n]$ where $f[m,n] = f(mD_x, nD_y)$. The ideal model for sampling considers multiplication of $f(x,y)$ by a grid of delta functions $s(x,y)$ where

$$s(x,y) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \delta(x - n_1 D_x, y - n_2 D_y) \tag{1}$$

where D_x, D_y are the separations between points on the sampling grid i.e. the *period* of horizontal and vertical sampling. Sampling (or digitization) is modelled as $f_s(x,y) = f(x,y)s(x,y)$. Now we 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 δ 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.

The derivation is similar to the 1D sampling theorem in 3C1. We start from

$$f_s(x, y) = f(x, y)s(x, y)$$

and realise that it becomes a convolution in the Fourier domain:

$$F_s(\omega_1, \omega_2) = F(\omega_1, \omega_2) \circledast S(\omega_1, \omega_2) \tag{2}$$

To compute the FT $S(\omega_1, \omega_2)$ of $s(x, y)$, it is convenient to realise that $s(x, y)$ is periodic and can be expressed as a Fourier series:

$$s(x, y) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} a_{k_1,k_2} e^{j(k_1 \omega_1^0 x + k_2 \omega_2^0 y)} \tag{3}$$

where $\omega_1^0 = \frac{2\pi}{D_x}$ and $\omega_2^0 = \frac{2\pi}{D_y}$ correspond to the fundamental frequency of $s(x, y)$ in both directions.

$$
\begin{aligned}
a_{k_1,k_2} &= \frac{1}{D_x D_y}\int_{-D_y/2}^{D_y/2}\int_{-D_x/2}^{D_x/2} s(x,y)e^{-j(k_1\omega_1^0 x+k_2\omega_2^0 y)}dxdy \\
&= \frac{1}{D_x D_y}\int_{-D_y/2}^{D_y/2}\int_{-D_x/2}^{D_x/2} \delta(x,y)e^{-j(k_1\omega_1^0 x+k_2\omega_2^0 y)}dxdy \\
&= \frac{1}{D_x D_y} \text{ By sifting property of the } \delta \text{ function}
\end{aligned} \tag{4}
$$

Therefore, substituting for a_{k_1,k_2} in equation 3 we can state that

$$
s(x,y) = \frac{1}{D_x D_y}\sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} e^{j(k_1\omega_1^0 x+k_2\omega_2^0 y)} \tag{5}
$$

and finally:

$$
S(\omega_1,\omega_2) = \frac{1}{D_x D_y}\sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} \delta\left(\omega_1 - k_1\omega_1^0, \omega_2 - k_2\omega_2^0\right) \tag{6}
$$

(cont.)

$$S(\omega_1, \omega_2) = \frac{1}{D_x D_y} \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \delta\left(\omega_1 - k_1 \omega_1^0, \omega_2 - k_2 \omega_2^0\right)$$

Thus, as

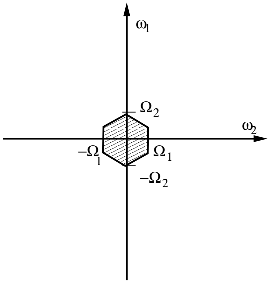
$$F_s(\omega_1, \omega_2) = F(\omega_1, \omega_2) \circledast S(\omega_1, \omega_2)$$

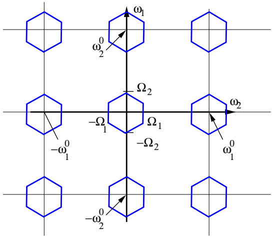
$$F_s(\omega_1, \omega_2) = \frac{1}{D_x D_y} \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} F(\omega_1 - k_1 \omega_1^0, \omega_2 - k_2 \omega_2^0) \tag{7}$$

Therefore sampling has the effect of scaling and replicating the spectrum.

2D Sampling Theorem

$F(\omega_1, \omega_2)$





$F_s(\omega_1, \omega_2)$

For no overlap in the spectrum 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 (it's similar to 1D).

$$\boxed{\omega_1^0 > 2\Omega_1 \quad \text{AND} \quad \omega_2^0 > 2\Omega_2} \tag{8}$$

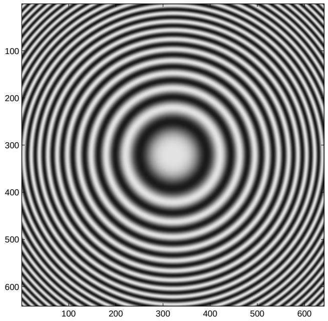
for no aliasing to occur. ($\omega_1^0 = 2\pi/D_x$ and $\omega_2^0 = 2\pi/D_y$)

2D Sampling Theorem

Perception of aliasing 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



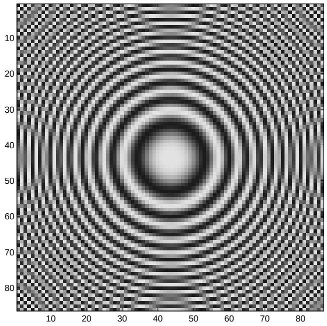
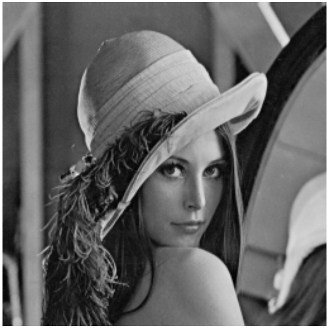
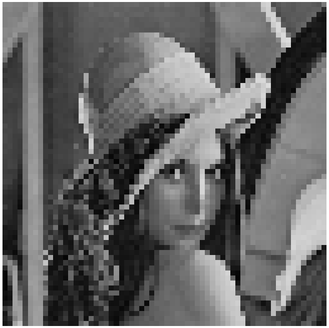


Figure: sinusoidal zone plate image $s(x, y) = \cos(a_x x^2 + a_y y^2)$, $a_x = b_x = 0.0628$

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

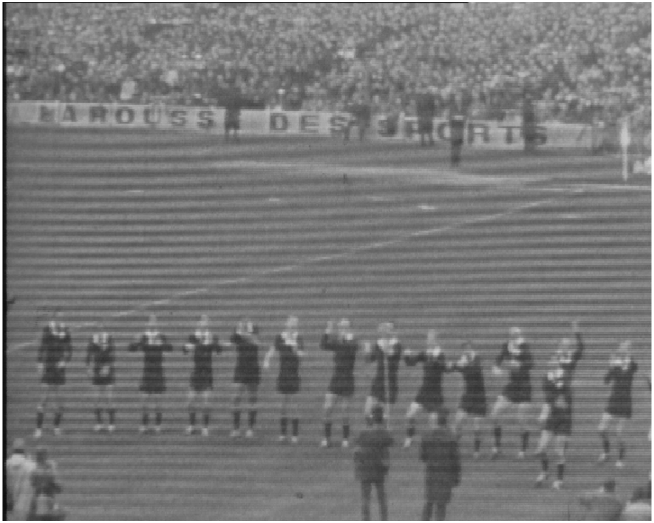
Lena





Left hand image sampled at $4\times$ 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 Transform

2D Discrete Fourier Transform

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

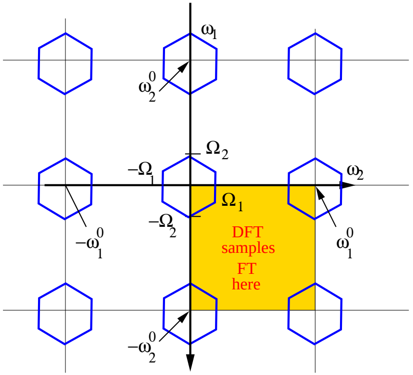
$$F[h,k] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n,m] e^{-j\left(\frac{2\pi}{N} nh + \frac{2\pi}{M} mk\right)}$$

$$f[n,m] = \frac{1}{NM} \sum_{h=0}^{N-1} \sum_{k=0}^{M-1} F[h,k] e^{j\left(\frac{2\pi}{N} nh + \frac{2\pi}{M} mk\right)} \tag{9}$$

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

Fast Algorithm exists (Lim pages 163–182). Tedious to derive, 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.

Symmetry Properties and other practical stuff



FFT calculates the spectral components at equal intervals up till the sampling frequency. Given $N \times M$ transform: $(0,0)^{th}$ bin is the DC coeff and N^{th} bin $\equiv 2\pi/D_y$ similarly for M^{th} bin.

Symmetry Properties and other practical stuff

Need to shuffle coefficients to get zero freq at centre. See what `fftshift` does.

Symmetry about DC: the DFT is centrally symmetric $F[h,k] = F^*[-h,-k]$.

The transform can be expressed as a matrix operation on rows then columns

$$F = W^T I W$$

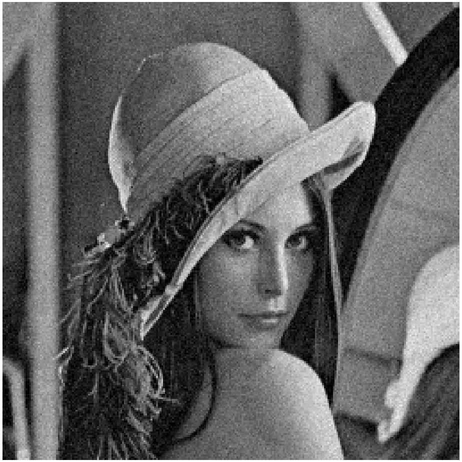
Filter Design in the Frequency Domain

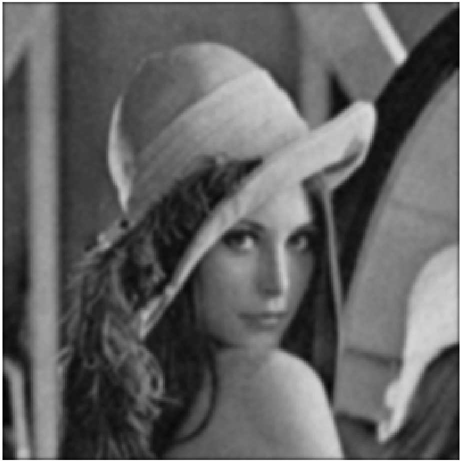
Can use 1D methods for separable filters.

It is possible to design FIR filters in the Frequency domain i.e. you can design in the frequency domain (continuous), take inverse then window and sample the impulse response.

Longer filters tend to run into problems due to the non-stationary statistics of images. For example, if we use a Low-Pass Filter to remove image noise, the edges will be blurred:

Filter Design in the Frequency Domain


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


Low Pass filter, 11×11 tap

In practice only short filters are possible if seeking real time performance.

2D Filter design is very important for wavelet analysis of signals. We will look later at the discrete wavelet transform for image compression (JPEG 2000).

For now, let's look at a simple example of filter design: Designing an-ideal low pass filter to prevent aliasing when downsampling an image.

Downsampling

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

We wish to implement factor of $\mathcal{N}$ downsampling in an image (decrease resolution by a factor of N). That is we want to take a discrete time signal $f[m,n]$ and create a downsampled signal:

$$f_{\mathcal{N}}[m,n] = f[m\mathcal{N}, n\mathcal{N}]$$

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

Recall (from sampling discussion) that the initial sampling rate is D_x, D_y pels per metre for the original sampled signal $f[m,n]$. Assuming the sampling rate is the same in both directions $D_x = D_y = D$ the sampling frequency is $[\Omega_0, \Omega_0] = [(2\pi/D), (2\pi/D)]$ radians/pixel. We 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 $f_{\mathcal{N}}[m,n]$.

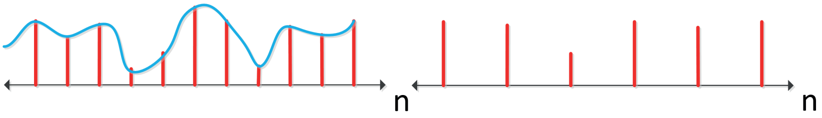


Figure: Equivalent problem in 1D. Left: An analogue signal is sampled at a sampling period of T. Right: The same signal downsampled by a factor of 2. Every second value is thrown away.

The spectrum of the sampled signal $f[m,n]$ contains (scaled) copies of the original spectrum $F(\omega_1,\omega_2)$ at integer multiples of $[\Omega_0,\Omega_0]$. If the sampling rate is reduced to $[\Omega_0/\mathcal{N},\Omega_0/\mathcal{N}]$ then the replicas are placed closer together in the signal spectrum.

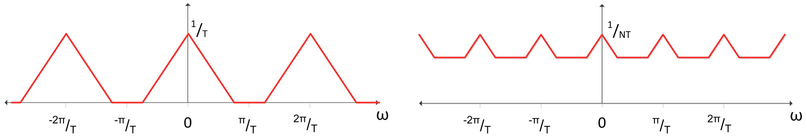
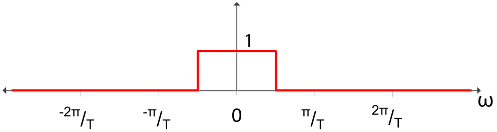


Figure: Left: The notional frequency Spectrum of the sampled signal. **Right**: The spectrum of the downsampled signal (by a factor of 2). As the signal contains frequencies greater than $\Omega_0/2N(=\pi/(2T)\ (N=2))$, aliasing will occur.

Thus, to prevent aliasing, we need to need to ensure that there are no spectral components above $[\Omega_0/(2\mathcal{N}), \Omega_0/(2\mathcal{N})]$, and apply a low-pass filter as follows:



Applying this low-pass filter, has the following effect on downsampling:

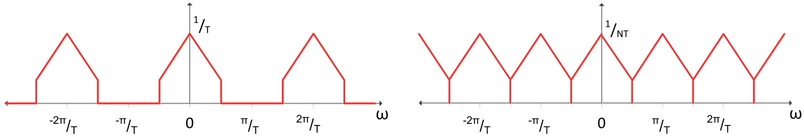


Figure: Left: The spectrum of the digital signal after an anti-aliasing filter is applied. **Right**: the downsampled signal spectrum after an anti-aliasing filter is applied.

Recall the effect of the original sampling at frequency Ω_0 which was used to obtain $f[m,n]$:

$$F_s(\omega_1,\omega_2) = \frac{1}{D^2}\sum_{k_1=-\infty}^{\infty}\sum_{k_2=-\infty}^{\infty} F(\omega_1 - k_1\Omega_0, \omega_2 - k_2\Omega_0)$$

We want to re-sample the original signal at a higher frequency. Thus we need to re-estimate the original spectrum $F(\omega_1,\omega_2)$. We can do this by cropping the replication at $(0,0)$ whilst cutting off the other replications, and then re-scaling its values.

This can be done with a low-pass filter $H_0(\omega_1,\omega_2) \rightleftarrows h(x,y)$ with frequency response as follows:

$$\begin{aligned} H_0(\omega_1,\omega_2) &= D^2\mathrm{rect}\left(\frac{\omega_1}{2\Omega_0}, \frac{\omega_2}{2\Omega_0}\right) \\ &= \begin{cases} D^2 & \text{for } |\omega_1| < \Omega_0 \text{ and } |\omega_2| < \Omega_0 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Simply applying $F(\omega_1, \omega_2) = H_0(\omega_1, \omega_2)F_s(\omega_1, \omega_2)$ gives us the spectrum of the analogue signal.

Now we want to re-sample the signal at $\Omega_0/\mathcal{N}$. To avoid aliasing in the re-sampling stage, we apply an ideal 2D anti-aliasing low-pass filter $H_{\mathcal{N}}(\omega_1, \omega_2)$ with cut-off at $[\pm\Omega_0/(2\mathcal{N}), \pm\Omega_0/(2\mathcal{N})]$. The frequency response is as follows:

$$\begin{aligned} H_{\mathcal{N}}(\omega_1, \omega_2) &= \mathrm{rect}\left(\frac{\omega_1 \mathcal{N}}{2\Omega_0}, \frac{\omega_2 \mathcal{N}}{2\Omega_0}\right) \\ &= \begin{cases} 1 & \text{for } |\omega_1| < \frac{\Omega_0}{\mathcal{N}} \text{ and } |\omega_2| < \frac{\Omega_0}{\mathcal{N}} \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Combining both effects 1) retrieving the original spectrum by cropping and rescaling and 2) low pass filtering the spectrum to avoid aliasing, we can derive our anti-aliasing filter $H(\omega_1, \omega_2)$ as follows:

$$\begin{aligned} H(\omega_1, \omega_2) &= H_0(\omega_1, \omega_2) H_{\mathcal{N}}(\omega_1, \omega_2) \\ &= D^2 \mathrm{rect}\left(\frac{\omega_1}{2\Omega_0}, \frac{\omega_2}{2\Omega_0}\right) \mathrm{rect}\left(\frac{\omega_1 \mathcal{N}}{2\Omega_0}, \frac{\omega_2 \mathcal{N}}{2\Omega_0}\right) \\ &= D^2 \mathrm{rect}\left(\frac{\omega_1 \mathcal{N}}{2\Omega_0}, \frac{\omega_2 \mathcal{N}}{2\Omega_0}\right) \\ &= \begin{cases} D^2 & \text{for } |\omega_1| < \frac{\Omega_0}{\mathcal{N}} \text{ and } |\omega_2| < \frac{\Omega_0}{\mathcal{N}} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{10}$$

The filter $H(\omega_1, \omega_2)$ is a 2D rectangular function in the frequency domain (horizontal and vertical bandwidth of $\Omega = \Omega_0/(2\mathcal{N})$). We've seen that in part 1. Using the Inverse 2D Fourier Transform:

$$\begin{aligned} h(x,y) &= \frac{1}{4\pi^2} \int\int D^2 \mathrm{rect}\left(\frac{\omega_1}{2\Omega}, \frac{\omega_2}{2\Omega}\right) e^{j(\omega_1 x + \omega_2 y)} d\omega_1 d\omega_2 \\ &= \times \frac{D^2}{4\pi^2} \int_{-\Omega}^{\Omega} \int_{-\Omega}^{\Omega} e^{j(\omega_1 x + \omega_2 y)} d\omega_1 d\omega_2 \\ &= \times \frac{D^2}{4\pi^2} \left(\frac{2}{x} \sin(x\Omega)\right) \int_{-\Omega}^{\Omega} e^{j(\omega_1 y)} d\omega_2 \\ &= \times \frac{D^2}{4\pi^2} \frac{2}{x} \sin(x\Omega) \frac{2}{y} \sin(y\Omega) \\ &= \frac{D^2}{4\pi^2} 2\Omega \mathrm{sinc}(x\Omega) 2\Omega \mathrm{sinc}(y\Omega) \end{aligned} \tag{11}$$

The filter is separable!

$$h(x,y) = \frac{D^2}{4\pi^2} 2\Omega \mathrm{sinc}(x\Omega) 2\Omega \mathrm{sinc}(y\Omega)$$

But need to apply filter in digital domain. Recall samples are at $y = hD$, $x = kD$ and that $\Omega = \Omega_0/2\mathcal{N} = 2\pi/(2D\mathcal{N}) = \pi/(D\mathcal{N})$:

$$\begin{aligned} h[m,n] &= D^2 \times \frac{1}{4\pi^2}\left[\frac{2\pi}{D\mathcal{N}}\mathrm{sinc}\left(\frac{mD\pi}{D\mathcal{N}}\right)\right]\left[\frac{2\pi}{D\mathcal{N}}\mathrm{sinc}\left(\frac{nD\pi}{D\mathcal{N}}\right)\right] \\ &= \left[\frac{1}{\mathcal{N}}\mathrm{sinc}\left(\frac{m\pi}{\mathcal{N}}\right)\right]\left[\frac{1}{\mathcal{N}}\mathrm{sinc}\left(\frac{n\pi}{\mathcal{N}}\right)\right] \end{aligned}$$

Importantly the filter does not rely on the original sampling frequency Ω_0, only on the sub-sampling factor $\mathcal{N}$.

Using Windowing to create an FIR Filter

This sinc function is infinitely long and so the filter is IIR. So it is necessary to truncate the filter to a finite length of N. However, simply trowing away values of the signal for $|k| > N/2$ will result in a ripple in the passband and stopband of the filter.

So need to multiply impulse response by some data window $w[m,n]$. Use separable hamming window $w[m]w[n]$. Thus

$$h[m,n] = \left[\frac{w[m]}{\mathcal{N}}\mathrm{sinc}\left(\frac{m\pi}{\mathcal{N}}\right)\right]\left[\frac{w[n]}{\mathcal{N}}\mathrm{sinc}\left(\frac{n\pi}{\mathcal{N}}\right)\right]$$

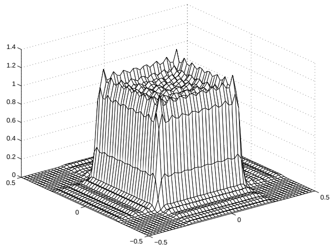
where

$$w[k] = \begin{cases} 0.54 - 0.46\cos(\frac{\pi k}{N}) & |k| < N/2 \\ 0 & \text{Otherwise} \end{cases} \tag{12}$$

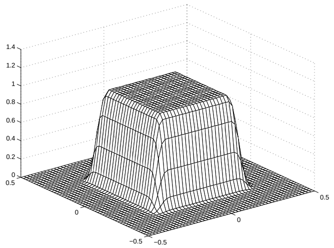
See Lim Chapter 4 for FIR Filter Design and better design method: frequency transformation pages 218–238. Also Applied Digital Signal Processing by Manolakis and Ingle has a good description of windowing and downsampling (1D).

Implementation

Gain of Frequency Response using a Rectangular Window



Gain of Frequency Response using a Hamming Window



The ripples in the passband and stopband are clearly visible when the IIR filter is simply truncated to form an FIR filter (FIR). When using the hamming window, the roll-off rate between the passband and stopband is sacrificed in order to remove the ripple.

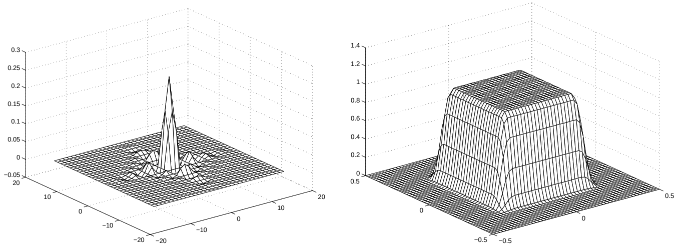
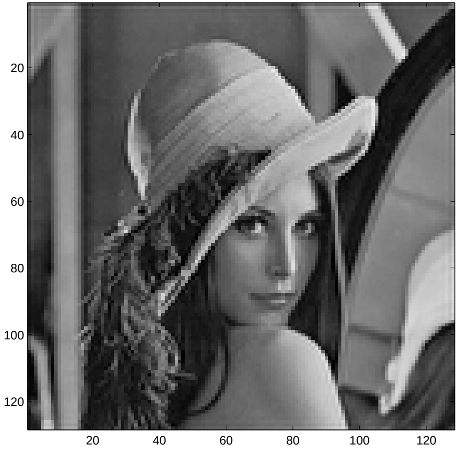


Figure: $h(x, y)$ and $|H(\omega_1, \omega_2)|$

```
N = 2;                     % subsampling factor
n = (-16:16);              % taps
h = (1/(N))*sinc(n/N);     % normalised sinc function = sinc(pi
w = hamming(length(n))'; % set window
h = h.*w;                  % window the impulse response
h = h/sum(sum(h));         % Make sure that DC gain of h is 1.0
h2D = h'*h;                % Make 2D filter from 1D filters.
                           % This is the same as h2D=conv2(h',h)
```

Implementation

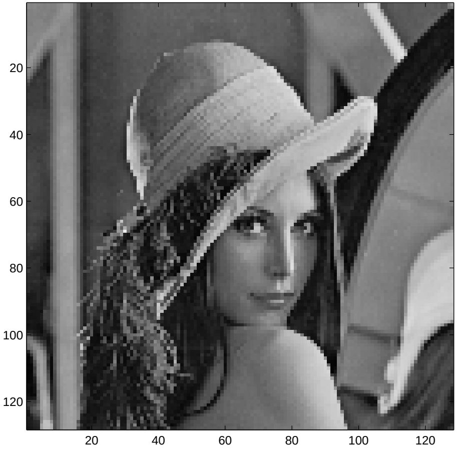
Original



subsampled $\mathcal{N} = 2$ with anti-aliasing

Implementation

Original



subsampled $\mathcal{N} = 2$ without anti-aliasing

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 a Gaussian shaped filter instead (also separable). Results acceptable for many applications.

$$h[m,n] = \exp - \left(\frac{m^2 + n^2}{2\sigma^2} \right) \tag{13}$$

where σ affects the filter bandwidth. Bigger σ implies bigger bandwidth. Need to specify the number of taps and to normalise the filter coeffs so that they sum to 1.0. `h = h/sum(sum(h))`.

we can get away with quite a short filter (15 taps).

BTW Good zooming or upsampling (needed for deinterlacing or TV→Cinema conversion) is *much* harder.

Filter Design & Examples

Application: Filters for Image Denoising (Low pass vs. Wiener)

We want to design a filter to remove noise from an image. We make the assumption that noise is Additive White Gaussian Noise.

Given an observed signal

$$g[m,n] = f[m,n] + \eta[m,n] \tag{14}$$

where $f[m,n]$ is the clean original and $\eta[m,n] \sim \mathcal{N}(0, \sigma_\eta^2)$; how do we recover $f[m,n]$ from $g[m,n]$?

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 like the gaussian or sinc filter to reduce the high-freq content of $g[m,n]$ should do the trick.

But that kills image features like edges and so causes a blurred image appearance.

The Wiener filter can be used to remove noise from the image. The frequency response of a filter can be defined in terms of the PSD of the signal $f(\cdot)$ and noise $\eta(\cdot)$ as follows

$$\begin{aligned} H(\omega_1, \omega_2) &= \frac{P_{gg}(\omega_1, \omega_2) - P_{\eta\eta}(\omega_1, \omega_2)}{P_{gg}(\omega_1, \omega_2)} \\ &= \frac{\text{PSD of Noisy Image} - \text{PSD of Noise}}{\text{PSD of Noisy Image}} \end{aligned} \tag{15}$$

Unlike the other filters the impulse response adapts to minimise the error between denoised image and the true clean image. However, it is necessary to know the PSD of both the observed image and the noise.

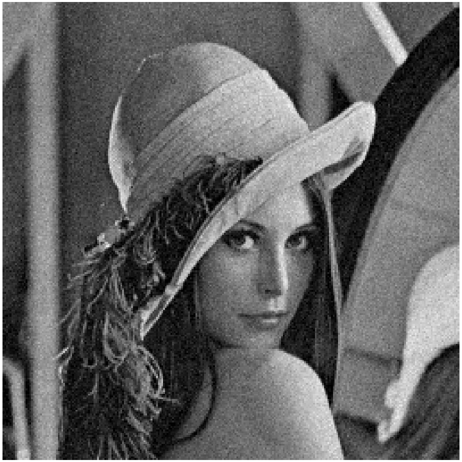
(Wiener filter cont.)

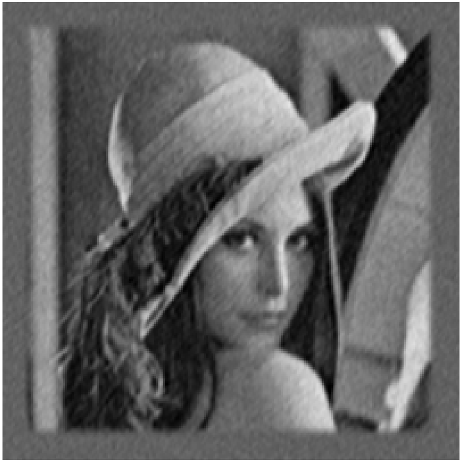
$$\begin{aligned} H(\omega_1, \omega_2) &= \frac{P_{gg}(\omega_1, \omega_2) - P_{\eta\eta}(\omega_1, \omega_2)}{P_{gg}(\omega_1, \omega_2)} \\ &= \frac{\text{PSD of Noisy Image} - \text{PSD of Noise}}{\text{PSD of Noisy Image}} \end{aligned}$$

Since η is white Gaussian noise, its PSD is a constant across the whole spectrum. This constant value $P_{\eta\eta}(\omega_1, \omega_2) \propto \sigma_\eta^2$. So all we need to do is estimate a value of σ_η^2 (possibly by trial and error).

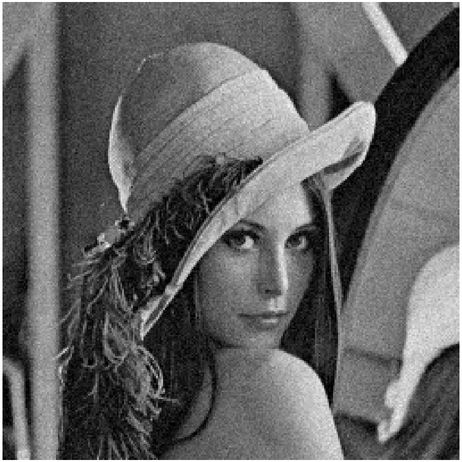
Can simulate the PSD of the observed image $(g(\cdot))$ by calculating $|\text{DFT}|^2$ in Matlab. BUT to estimate the DFT of $g(\cdot)$ the data needs to be windowed with some analysis window to prevent spectral leakage (eg. 2D Hamming).

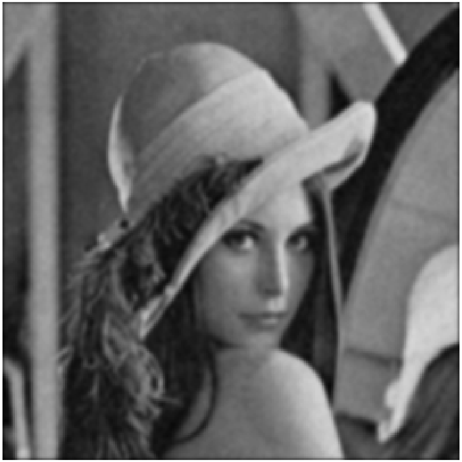
Application: Noise Reduction (Low pass vs. Wiener)


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


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

Application: Noise Reduction (Low pass vs. Wiener)


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

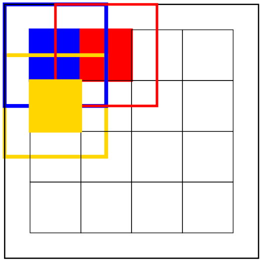

Gaussian filter, 11×11 tap $\sigma^2_h = \sigma^2_k = 1.5$

Implementation: Block based image processing

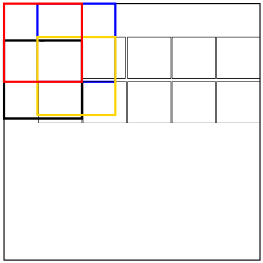
The result is unsatisfactory. The problem is that underlying image is statistically non-stationary.

Over smaller areas the image can be assumed to be approximately stationary. This assumption can be used to make the wiener filter more effective.

The idea is to filter blocks separately then either tile the output or overlap the output.

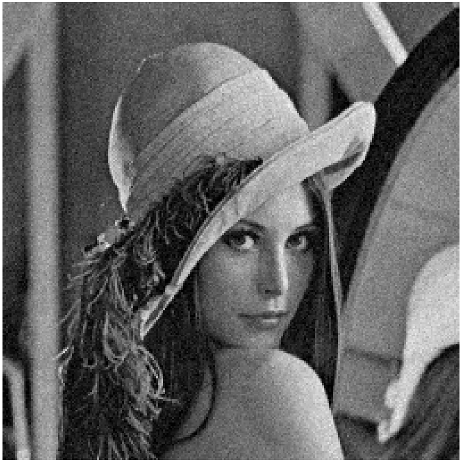


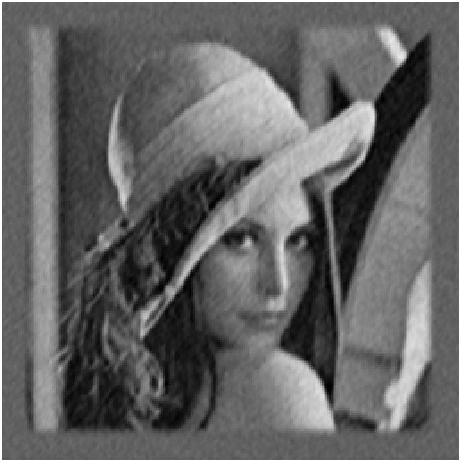
Left: Overlapped Tiling (overlap/save).



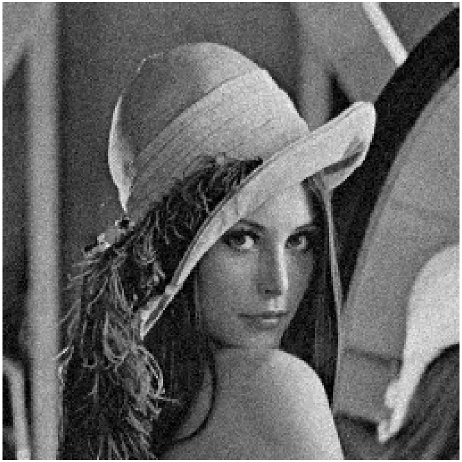
Right: Overlap/add (2:1 overlap)

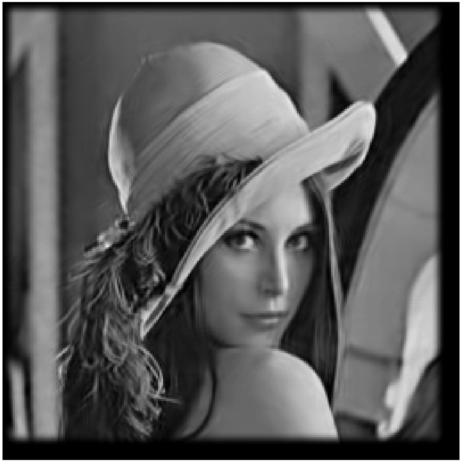
Application: Noise Reduction (Low pass vs. Wiener)


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

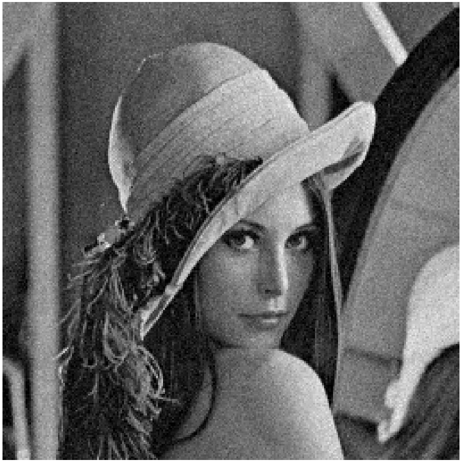

Wiener result, Hamming analysis window using $\sigma_{\eta\eta}^2 = 500$

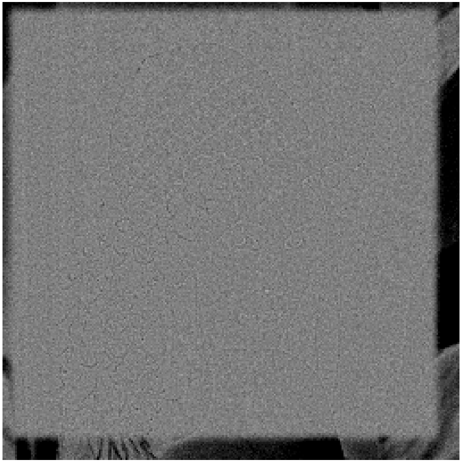
Application: Noise Reduction (Low pass vs. Wiener)


Noisy Lena $\boldsymbol{\sigma_{\eta\eta}^2 = 100}$ about 28dB SNR


Wiener filter applied on blocks (overlap/add)

Application: Noise Reduction (Low pass vs. Wiener)

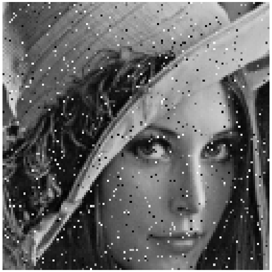

Noisy Lena $\sigma_{\eta\eta}^2 = 100$ about 28dB SNR

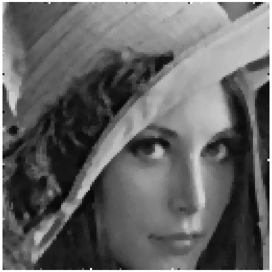

Difference of Wiener filter applied on blocks with noisy original

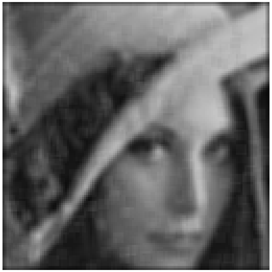
Application: Impulse noise reduction using a non-linear filter

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


5×5 median filter


5×5 average filter

Impulse noise reduction using a non-linear filter

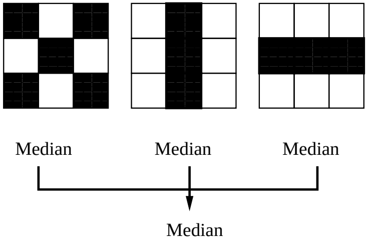
This filter is one from the class of *Order statistic*.

The median filter is quite slow because a sort is needed to calculate the output at each pixel.

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

Looked at sampling of 2D continuous signals and factor of $\mathcal{N}$ down-sampling of discrete signals.

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