COMP9517

Computer Vision

2023 Term 2 Week 2

Professor Erik Meijering

Image Processing

Part 2-2

UNSW SYDNEY

Types of image processing (recap)

- Two main types of image processing operations:
  - **Spatial domain operations** (in image space)
  - **Transform domain operations** (mainly in Fourier space)

Today

- Two main types of spatial domain operations:
  - **Point operations** (intensity transformations on individual pixels)
  - **Neighbourhood operations** (spatial filtering on groups of pixels)

Topics and learning goals

- Describe the principles of the **Fourier transform for image processing**

  Forward & inverse transform, convolution theorem, properties, discrete Fourier transform

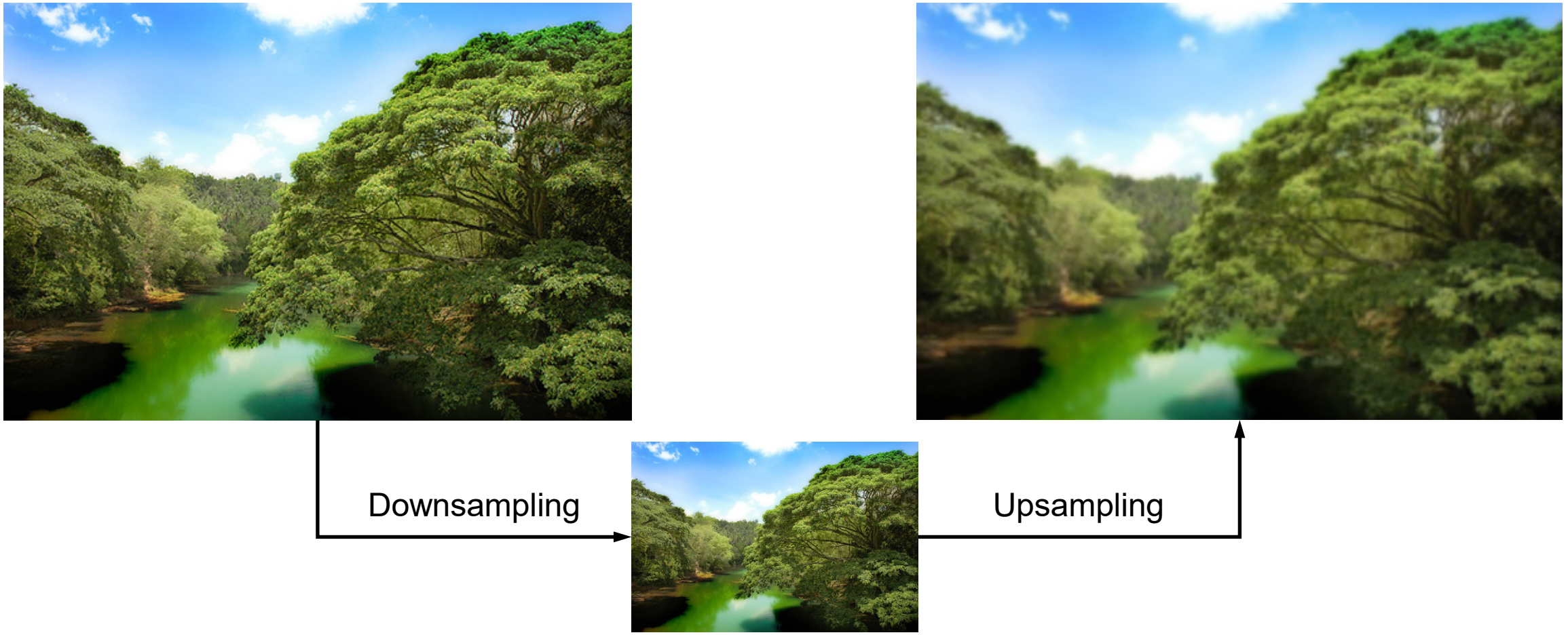
- Understand the effects of various **Fourier domain filtering methods**

  Filtering procedure, notch filtering, low-pass filtering, high-pass filtering

- Combine filtering operations to allow **multiresolution image processing**

  Difference of Gaussians, image pyramids, approximation, reconstruction

What is lost when lowering resolution?



Jean Baptiste Joseph Fourier (1768-1830)

Had a crazy idea (1807)

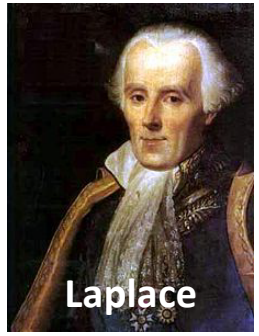
Any univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies

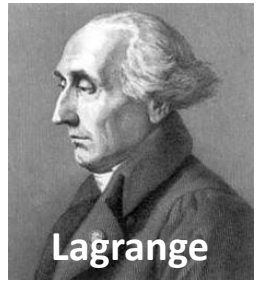
Don't believe it?

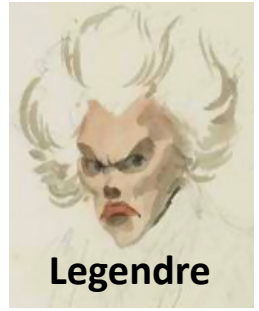
- Neither did Lagrange, Laplace, Legendre, and other big wigs
- Fourier's idea was not translated into English until 1878

But it's (mostly) true!

- It is now called the Fourier series
- There are some subtle restrictions







"...the manner in which the author arrives at these equations is not exempt of difficulties and...his analysis to integrate them still leaves something to be desired on the score of generality and even rigour."

Weighted sum of sines

- **Basic building block**

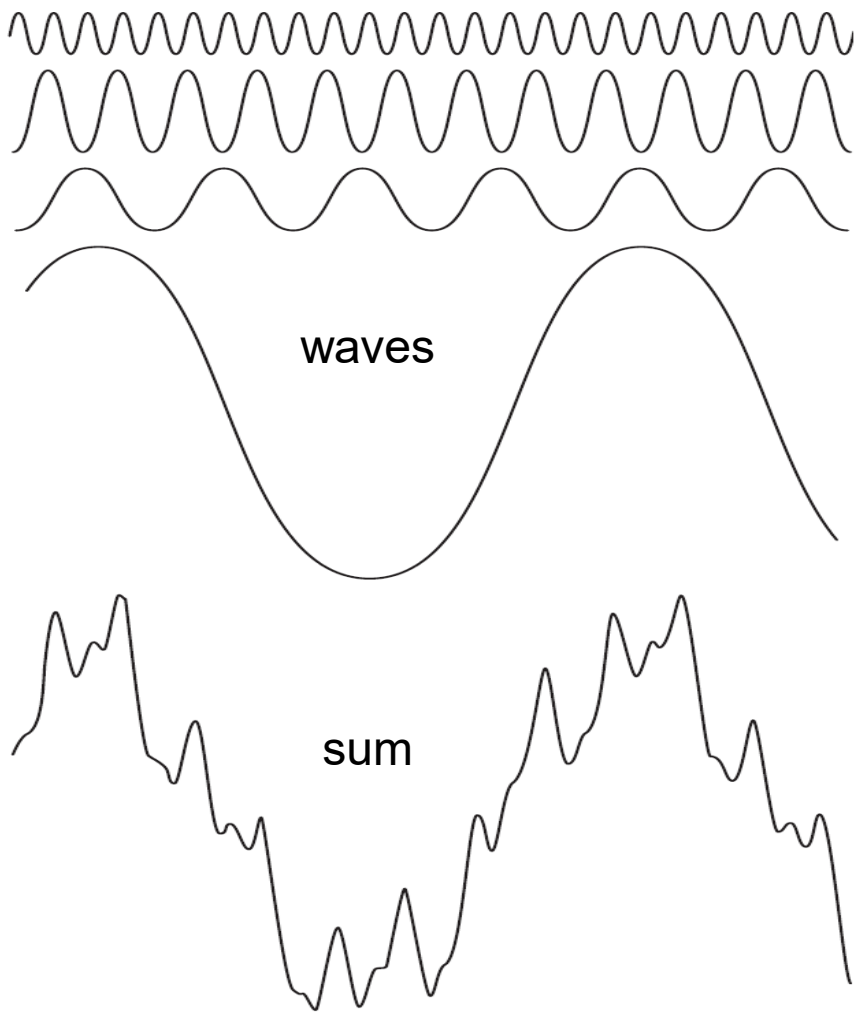
  $f_i(x) = a_i \sin(\omega_i x + \varphi_i)$

  a_i is the weight (amplitude)

  ω_i is the radial frequency

  φ_i is the phase

Add enough of them and you can get any signal you want: $f = f_0 + f_1 + f_2 + f_3 + \cdots$



Weighted sum of sines

https://en.wikipedia.org/wiki/Fourier_series

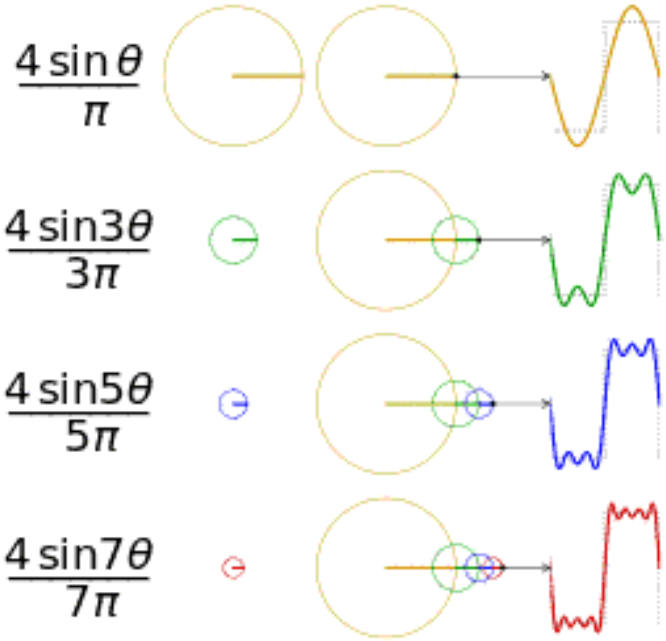
f_1

f_2

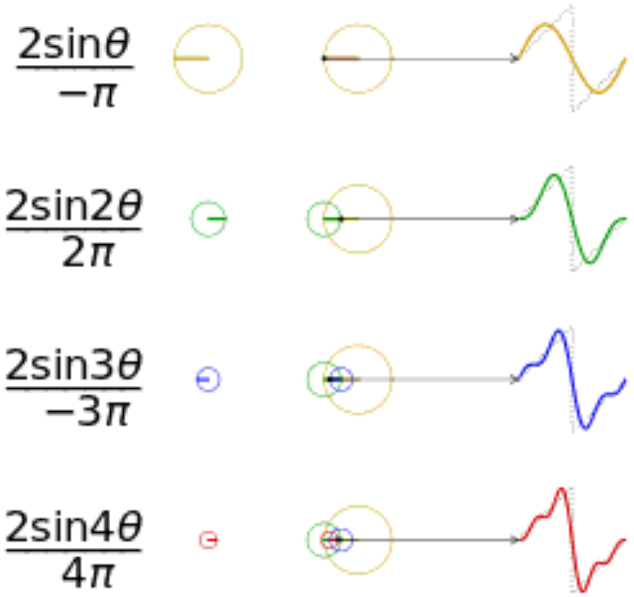
f_3

f_4

$\vdots$



Approximation of a square wave



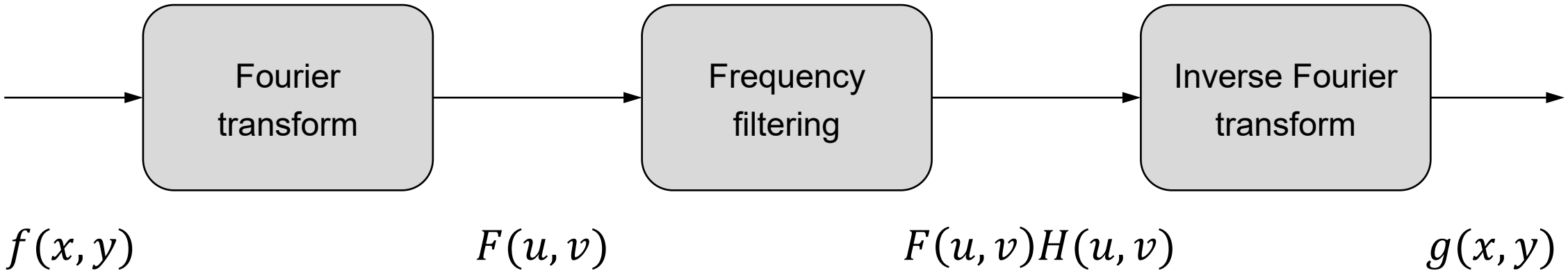
Approximation of a sawtooth wave

Spatial versus frequency domain

- **Spatial domain**
  - The image plane itself
  - Direct manipulation of pixels
  - Changes in pixel position correspond to changes in the scene
- **Frequency domain**
  - Fourier transform of an image
  - Directly related to rate of changes in the image
  - Changes in pixel position correspond to changes in the frequency

Frequency domain overview

- High frequencies correspond to rapidly changing intensities across pixel
- Low frequency components correspond to large-scale image structures
- Frequency domain image processing via the Fourier transform



Definition of the Fourier transform (1D)

- **Forward Fourier transform**

$$F(u) = \int_{-\infty}^{\infty} f(x) \mathrm{e}^{-i2\pi ux} dx$$

- **Inverse Fourier transform**

$$f(x) = \int_{-\infty}^{\infty} F(u) \mathrm{e}^{i2\pi ux} du$$

Uses complex valued sinusoids

Notation

$f(x)$ is the spatial input function

$F(u)$ is the Fourier transform

$\mathrm{e}^{i\omega x} = \cos(\omega x) + i \sin(\omega x)$

$\omega = 2\pi u$ is radial frequency

u is spatial frequency

$i = \sqrt{-1}$

Properties of the Fourier transform

Property	Spatial	Frequency
Superposition	$f_1(x) + f_2(x)$	$F_1(u) + F_2(u)$
Translation	$f(x - \Delta x)$	$F(u)\mathrm{e}^{-i2\pi u\Delta x}$
Convolution	$f(x) * h(x)$	$F(u)H(u)$
Correlation	$f(x) \otimes h(x)$	$F(u)H^*(u)$
Multiplication	$f(x)h(x)$	$F(u) * H(u)$
Scaling	$f(ax)$	$F(u/a)/a$
Differentiation	$f^{(n)}(x)$	$(i2\pi u)^n F(u)$

Definition of the Fourier transform (2D)

- **Forward Fourier transform**

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\mathrm{e}^{-i2\pi(ux+vy)}dxdy$$

- **Inverse Fourier transform**

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)\mathrm{e}^{i2\pi(ux+vy)}dudv$$

Uses complex valued sinusoids

Fourier transform pair

$$f \leftrightarrow F$$

$$F = R + i\,I$$

Real part + Imaginary part

Amplitude

$$a = \sqrt{R^2 + I^2}$$

Phase

$$\varphi = \tan^{-1}(I/R)$$

Discrete Fourier transform (DFT)

Digital images are discrete 2D functions

- **Forward discrete Fourier transform**

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\mathrm{e}^{-i2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

for $u = 0 \ldots M-1$ and $v = 0 \ldots N-1$

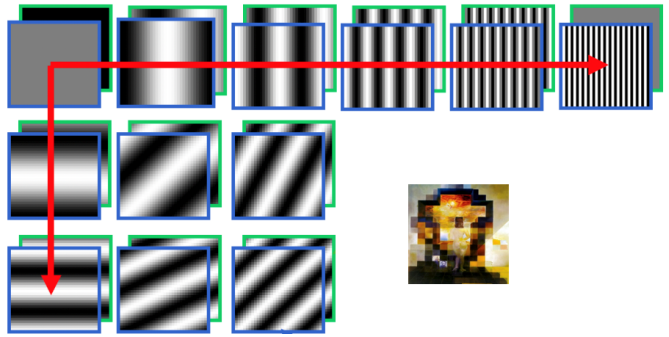
- **Inverse discrete Fourier transform**

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\mathrm{e}^{i2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

for $x = 0 \ldots M-1$ and $y = 0 \ldots N-1$

The discrete Fourier transform and its inverse always exist

Discrete Fourier transform basis

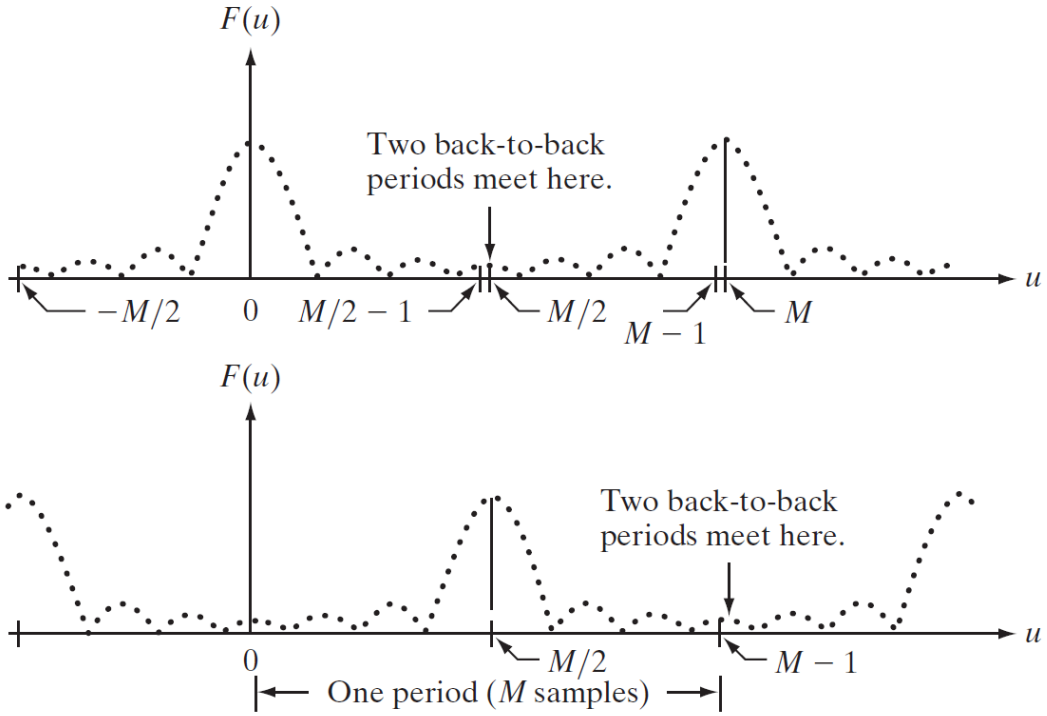


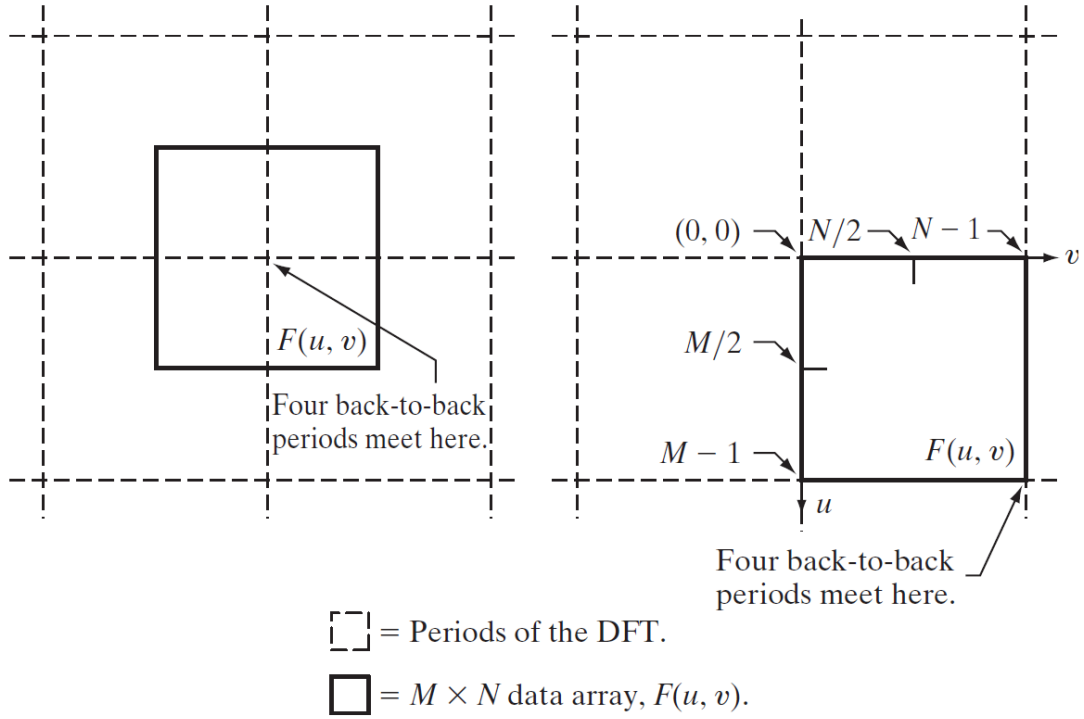
Frequency domain filtering

- Each $F(u,v)$ depends on all $f(x,y),\ x = 0 \ldots M-1,\ y = 0 \ldots N-1$
- Frequencies in the Fourier domain correspond to patterns in the image
- The value of $F(0,0)$ is the average intensity over all pixels of the image
- Low frequencies correspond to slowly varying intensities across pixels
- High frequencies correspond to rapid intensity changes across pixels
- Noise typically corresponds to fluctuations in the highest frequencies

Frequency domain filtering

- Fourier images are typically centred for visualisation and processing





Frequency domain filtering

- Centering the Fourier images means multiplying the spatial images by $(-1)^{x+y}$

Translation property: $F(u-u_0, v-v_0) \leftrightarrow f(x,y)\mathrm{e}^{i2\pi\left(\frac{u_0 x}{M}+\frac{v_0 y}{N}\right)}$

Centering by translation over: $u_0 = M/2$ and $v_0 = N/2$

Substitution yields: $F(u-M/2, v-N/2) \leftrightarrow f(x,y)\mathrm{e}^{i\pi(x+y)}$

Which boils down to: $f(x,y)\big(\cos\big(\pi(x+y)\big) + i\sin\big(\pi(x+y)\big)\big) = f(x,y)(-1)^{x+y}$

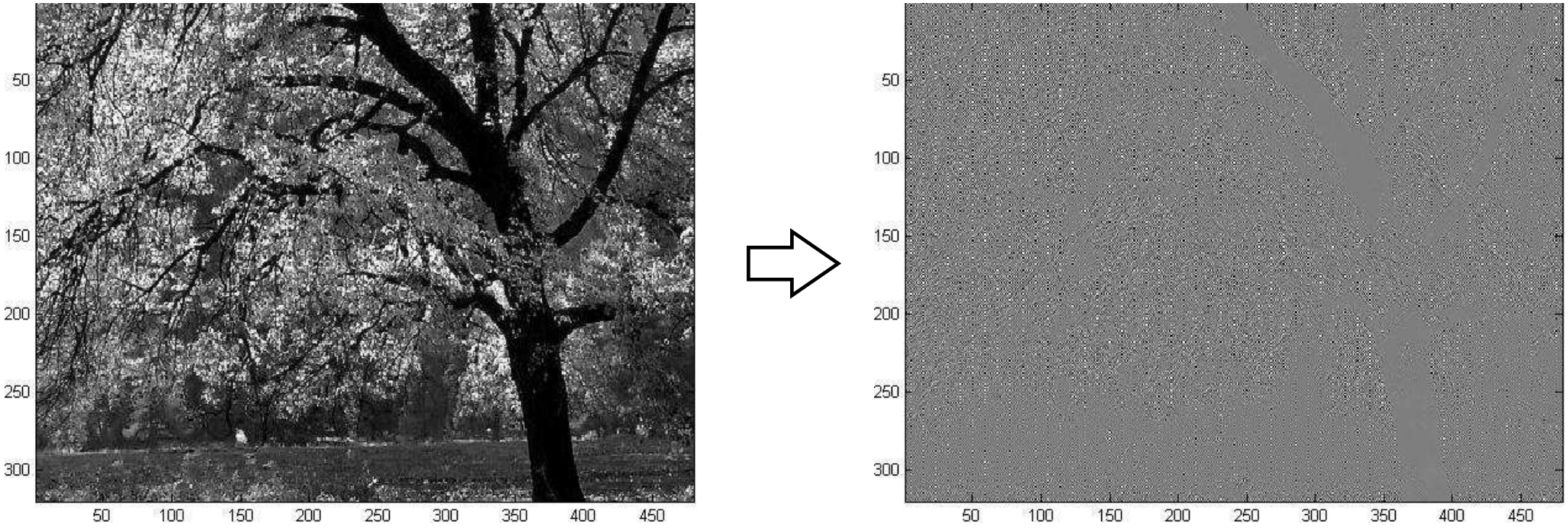
$= -1$ or 1 $= 0$ $x+y$ has integer value

Procedure for frequency domain filtering

1. Multiply the input image $f(x, y)$ by $(-1)^{x+y}$ to ensure centering $F(u, v)$
2. Compute the transform $F(u, v)$ from image $f(x, y)$ using the 2D DFT
3. Multiply $F(u, v)$ by a centred filter $H(u, v)$ to obtain the result $G(u, v)$
4. Compute the inverse DFT of $G(u, v)$ to obtain the spatial result $g(x, y)$
5. Take the real component of $g(x, y)$ (the imaginary component is zero)
6. Multiply the result by $(-1)^{x+y}$ to remove the pattern introduced in 1.

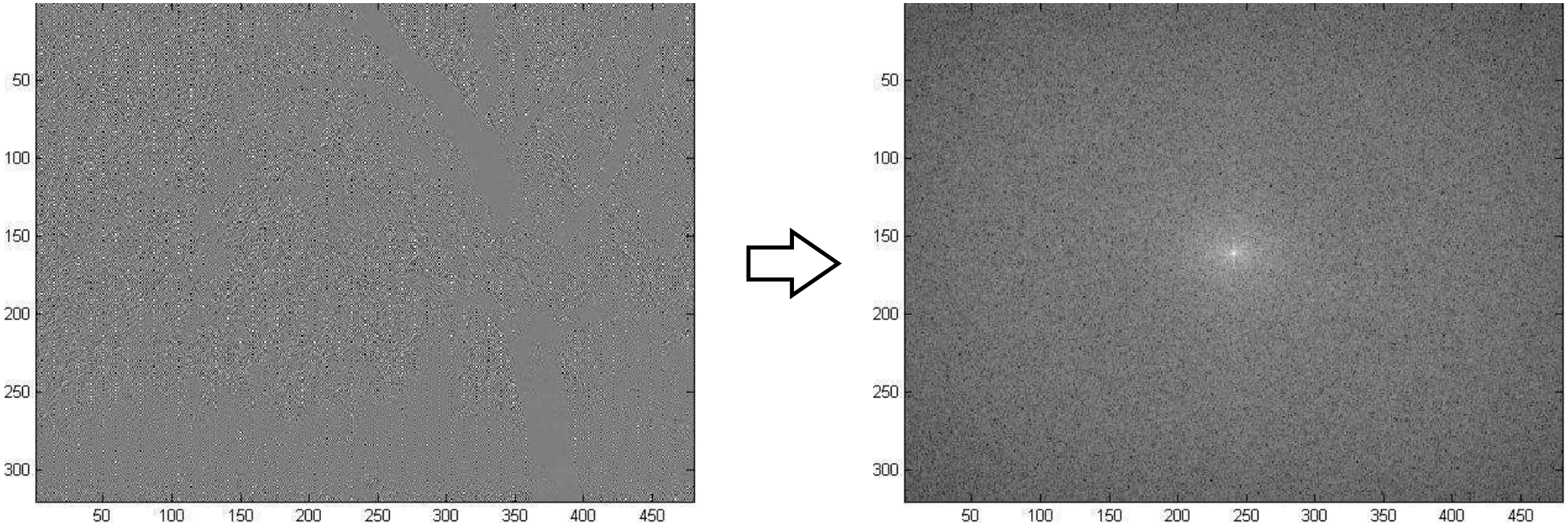
Example: low-pass filtering

1. Multiply the input image $f(x, y)$ by $(-1)^{x+y}$ to ensure centering $F(u, v)$



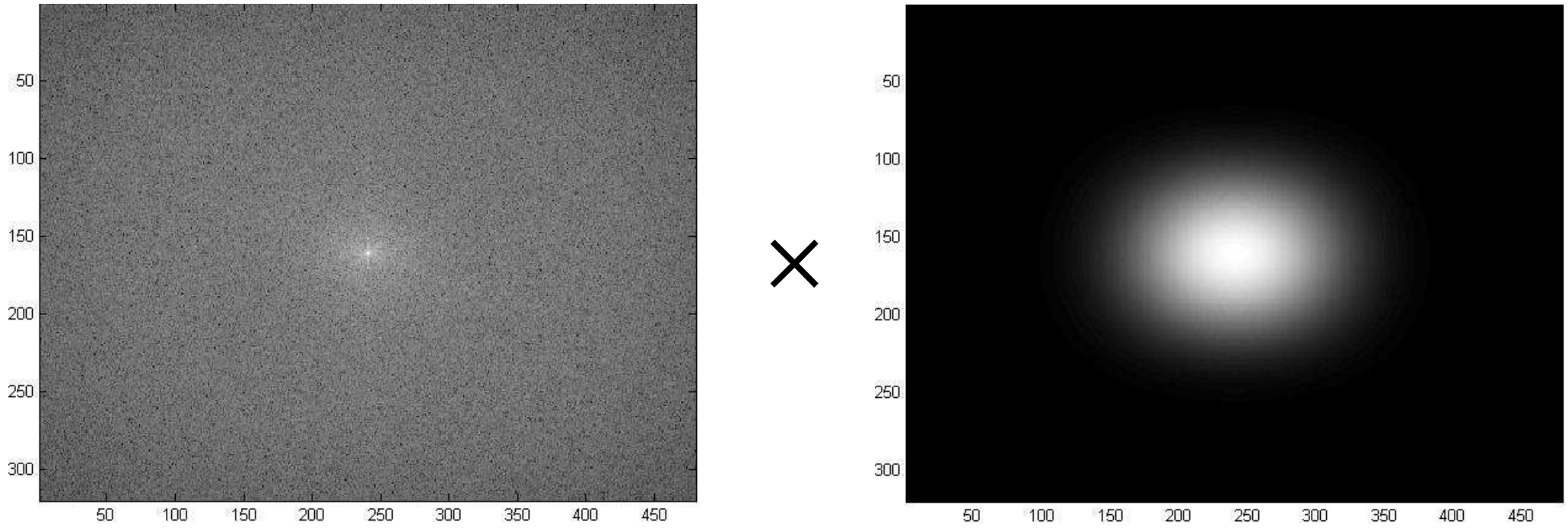
Example: low-pass filtering

2. Compute the transform $F(u, v)$ from image $f(x, y)$ using the 2D DFT



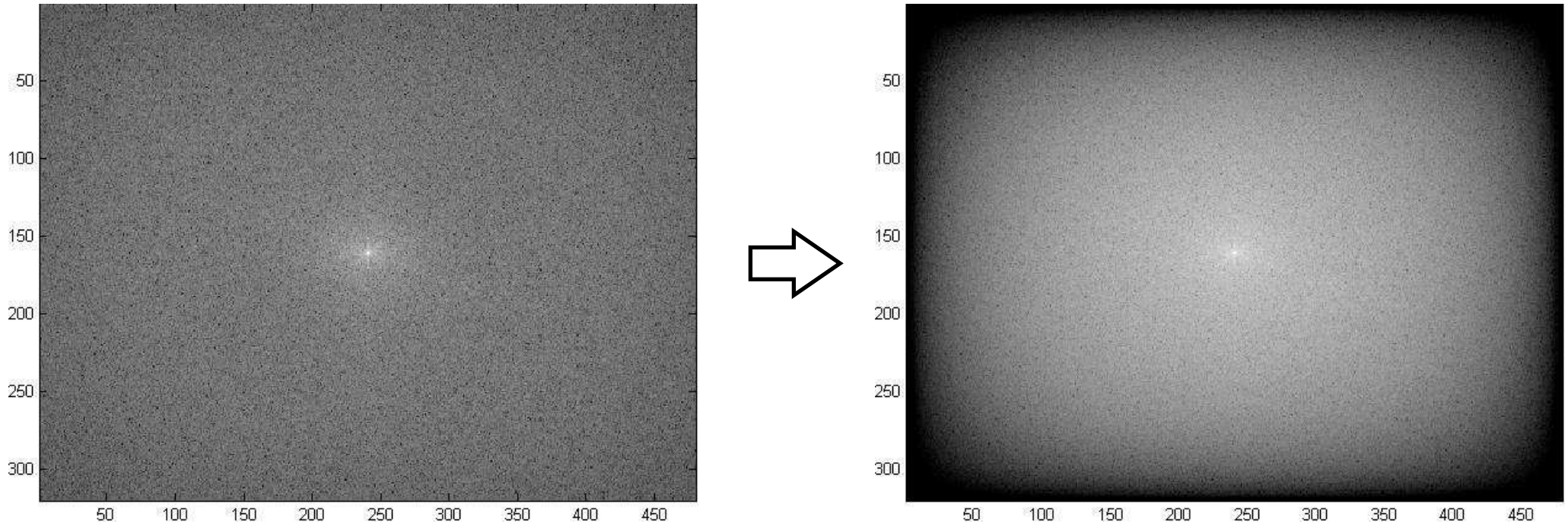
Example: low-pass filtering

3. Multiply $F(u,v)$ by a centred filter $H(u,v)$ to obtain the result $G(u,v)$



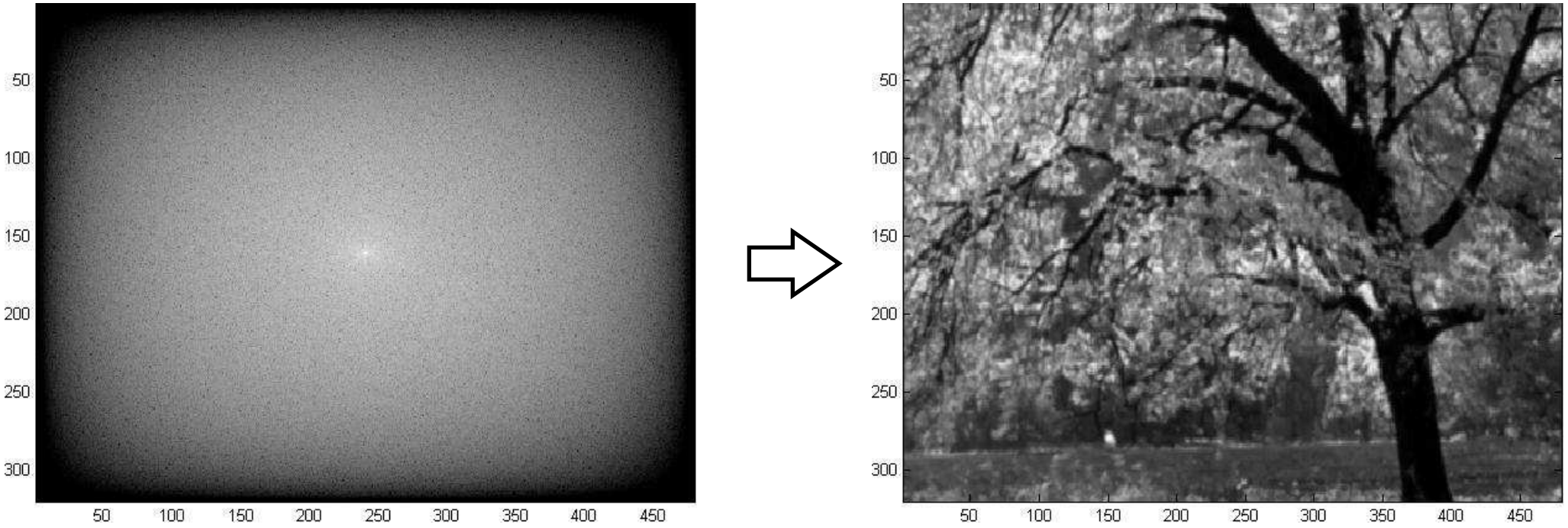
Example: low-pass filtering

3. Multiply $F(u,v)$ by a centred filter $H(u,v)$ to obtain the result $G(u,v)$



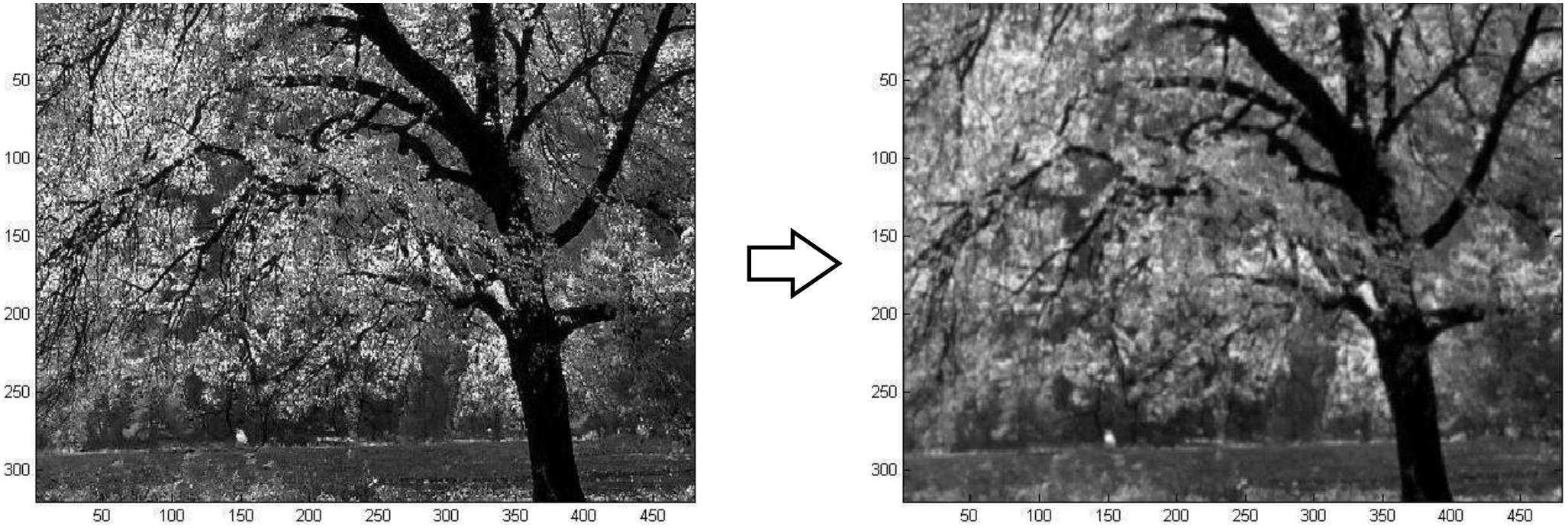
Example: low-pass filtering

4. Compute inverse DFT 5. Take real component 6. Multiply by $(-1)^{x+y}$



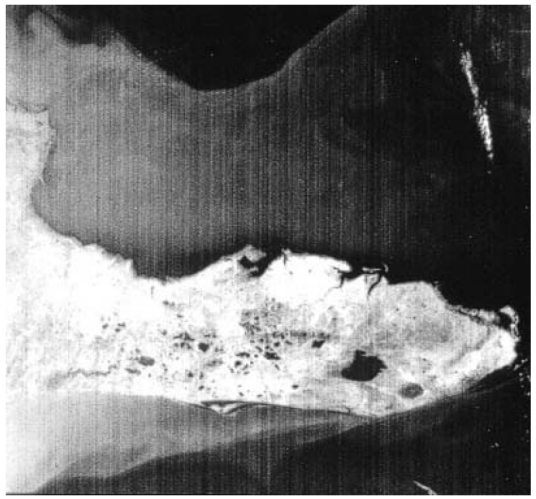
Example: low-pass filtering

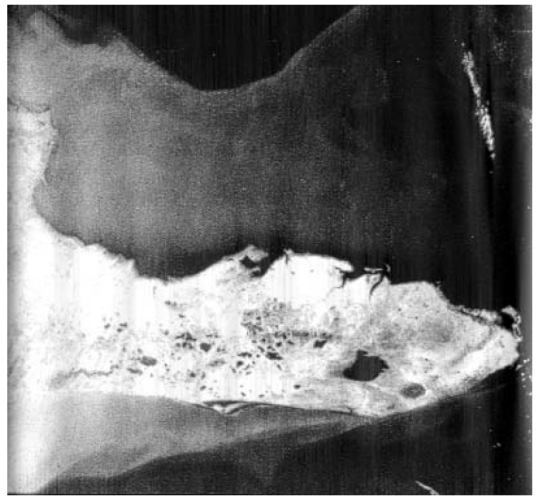
- The resulting low-pass filtered image is a smoothed version of the original



Example: notch filtering

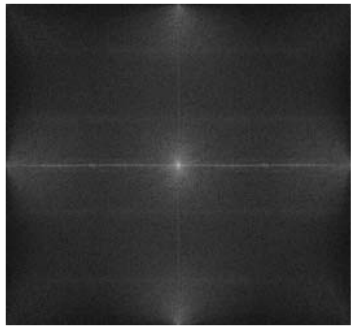
Satellite image showing scanline artifacts

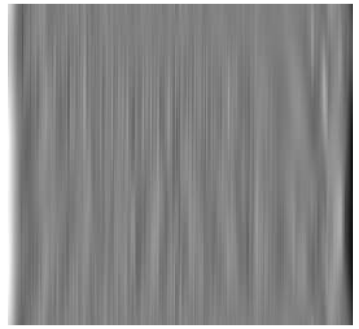




Result image using the **notch reject filter**

Fourier transform of the satellite image





Noise pattern captured by the **notch pass filter**

Notch pass filter

Exploiting the convolution theorem

- Filtering in the frequency domain can be computationally more efficient
- Designing filters can be more intuitive in the frequency domain
- Low-pass filter: keep low frequencies but attenuate high frequencies
- High-pass filter: keep high frequencies but attenuate low frequencies
- Band-pass filter: keep frequencies in a given band and attenuate the rest
- Take the inverse transform to get the corresponding spatial filter

Gaussian filter (low-pass)

- The Fourier transform of a Gaussian is a Gaussian

1D: $$h(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{x^2}{2\sigma^2}} \quad \leftrightarrow \quad H(u) = \mathrm{e}^{-2\pi^2\sigma^2u^2}$$

2D: $$h(x,y) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{x^2+y^2}{2\sigma^2}} \quad \leftrightarrow \quad H(u,v) = \mathrm{e}^{-2\pi^2\sigma^2(u^2+v^2)}$$

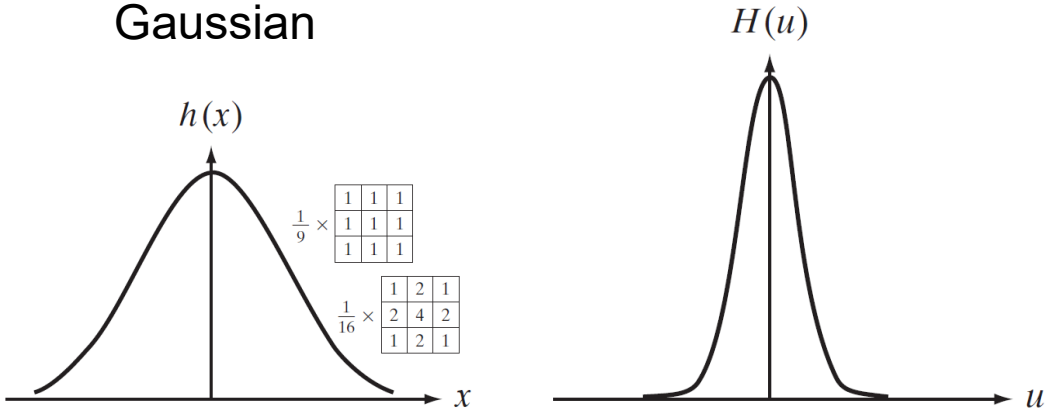
*n*D: $$h(x_1,\ldots,x_n) = \left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \mathrm{e}^{-\frac{x_1^2+\ldots+x_n^2}{2\sigma^2}} \quad \leftrightarrow \quad H(u_1,\ldots,u_n) = \mathrm{e}^{-2\pi^2\sigma^2(u_1^2+\ldots+u_n^2)}$$

Difference of Gaussian filter (high-pass)

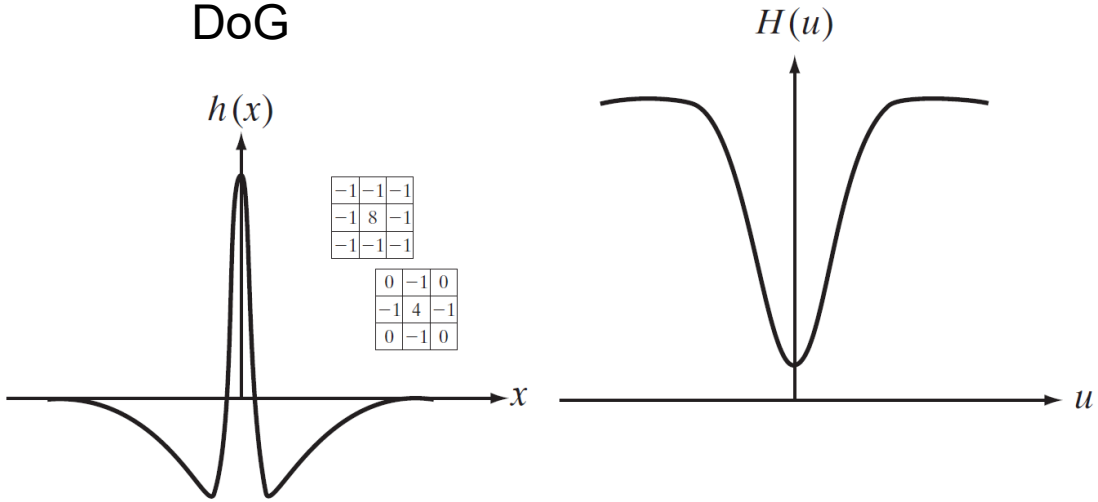
- Approximation of an inverted Laplacean filter

$$\mathrm{DoG}(x) = a_1 h(x;\sigma_1) - a_2 h(x;\sigma_2) \quad \leftrightarrow \quad \mathrm{DoG}(u,v) = a_1 H(u;\sigma_1) - a_2 H(u;\sigma_2)$$

Gaussian



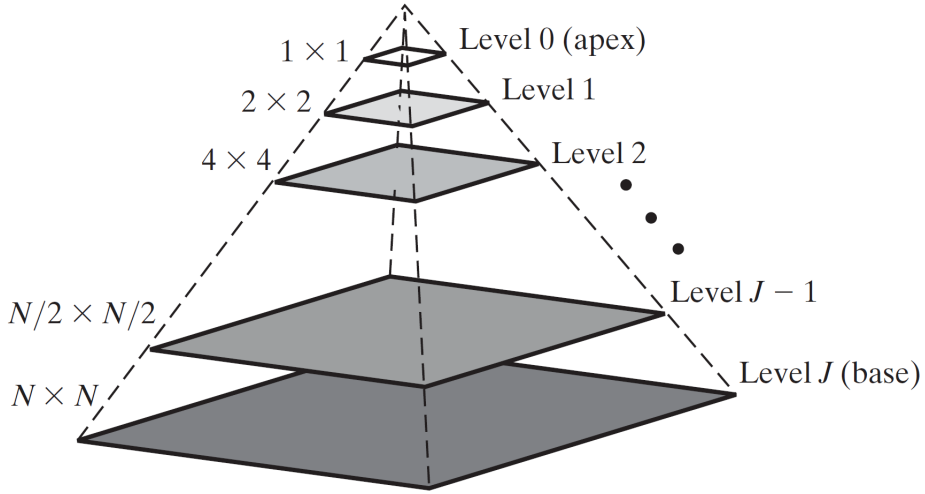
DoG



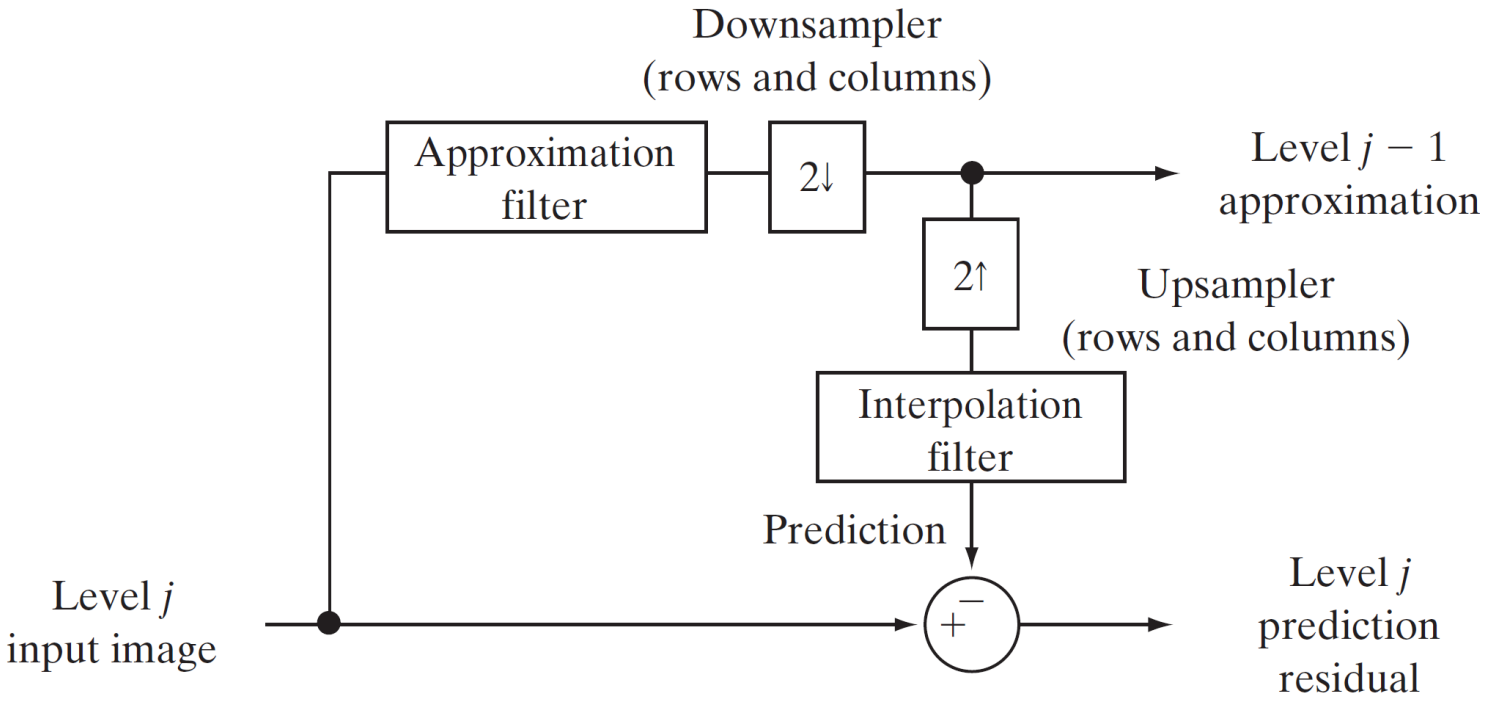
What is lost when lowering resolution?

Multiresolution image processing

- Small objects and fine details benefit from high resolution
- Large objects and coarse structures can make do with lower resolution
- If both are present at the same time, multiple resolutions may be useful
- This requires computing image pyramids



Creating image pyramids



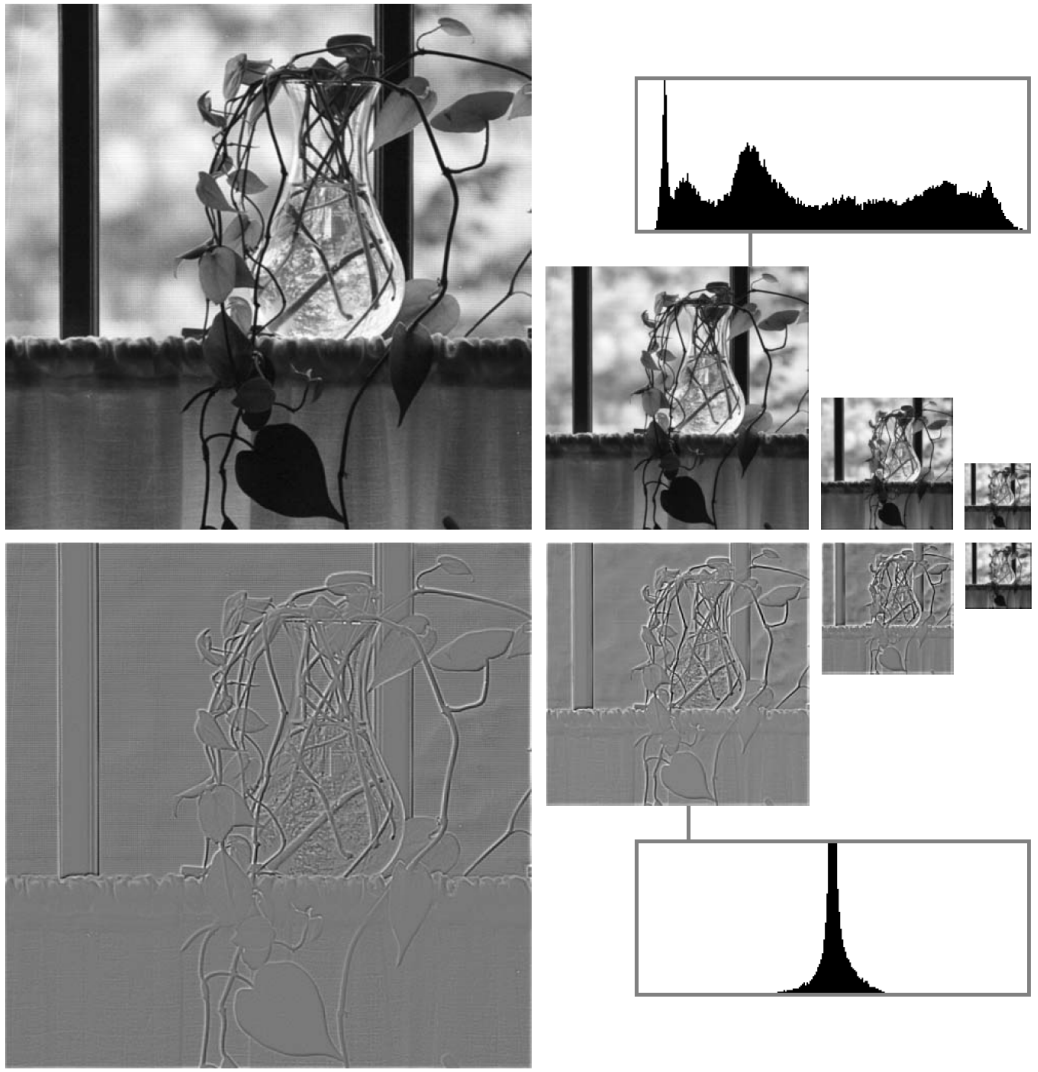
1. Compute an approximation of the input image by filtering and downsampling
2. Upsample the output of step 1 and filter the result (interpolation)
3. Compute the difference between the prediction of step 2 and the input to step 1

Repeating this produces an approximation and prediction residual pyramid

Example

Approximation pyramid

Residual pyramid



To reconstruct the image:

1. Upsample and filter the lowest resolution approximation image
2. Add the one-level higher prediction residual

Further reading on discussed topics

- Chapter 4 of Gonzalez and Woods 2002
- Sections 3.4-3.5 of Szeliski 2010

Acknowledgement

- Some images drawn from the above resources