— CM50248 — 2017/2018 —

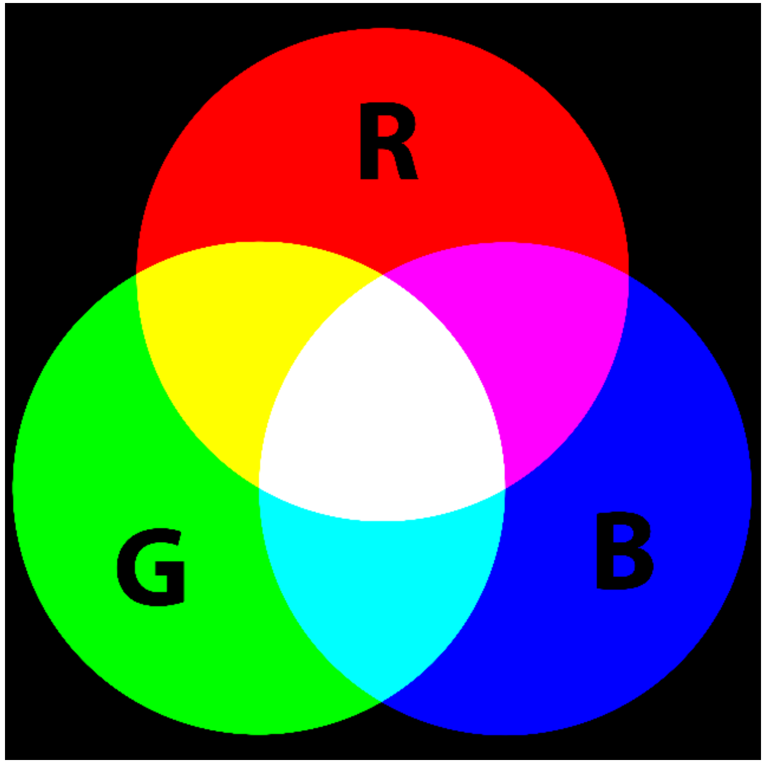
Visual Understanding 1

Dr Christian Richardt

Recap

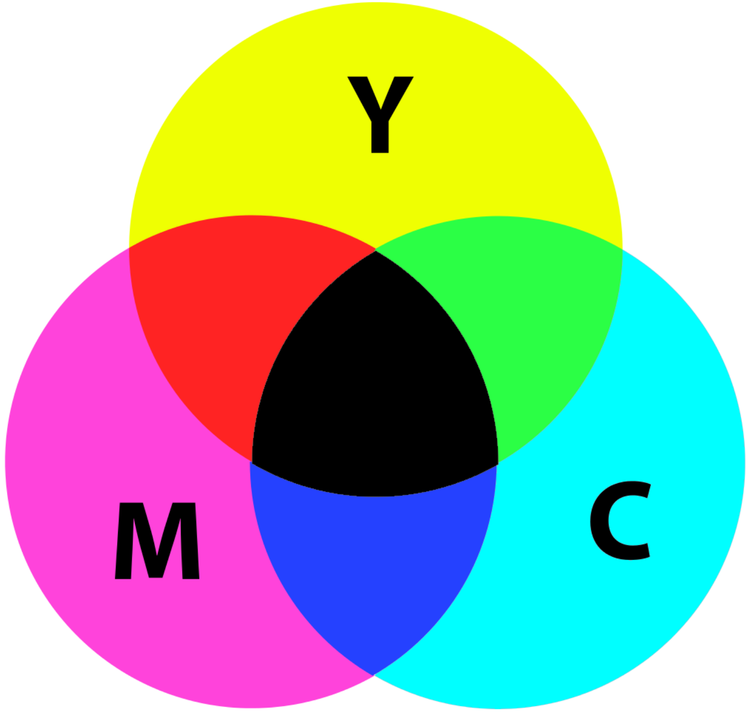
Additive and subtractive primary colours (recap)

RGB – additive



Mixing lights
e.g. LCD monitors

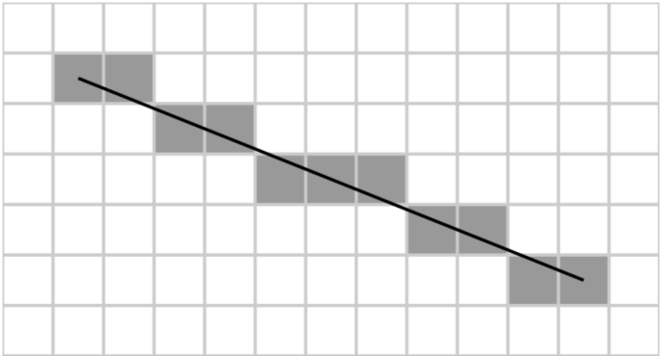
CMY – subtractive

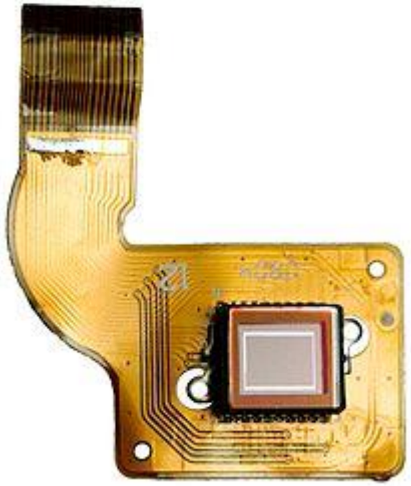


Filters and paints
e.g. printers

Digital image formation (recap)

- **Rasterisation**: convert a continuous or vector image representation to a rectangular sampled grid of pixels
- Rasterising vector graphics
  - e.g. Bresenham line drawing





- Sampling an analogue signal
  - e.g. in a digital camera
  - Digital cameras have a CCD or CMOS sensor with light sensitive elements, typically arranged in a “Bayer” pattern

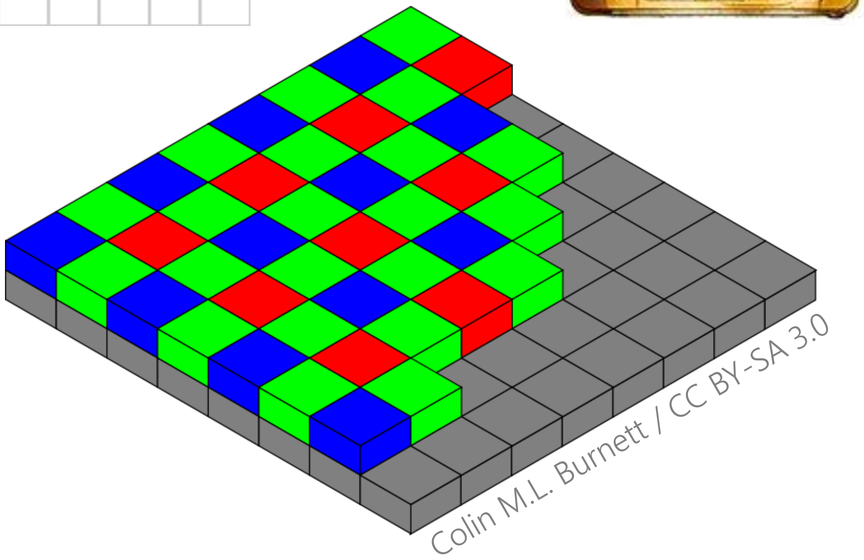


Image storage (greyscale, recap)

- Digital images are arranged in memory in scanline order:
  - pixels from left to right within a row
  - rows from top to bottom

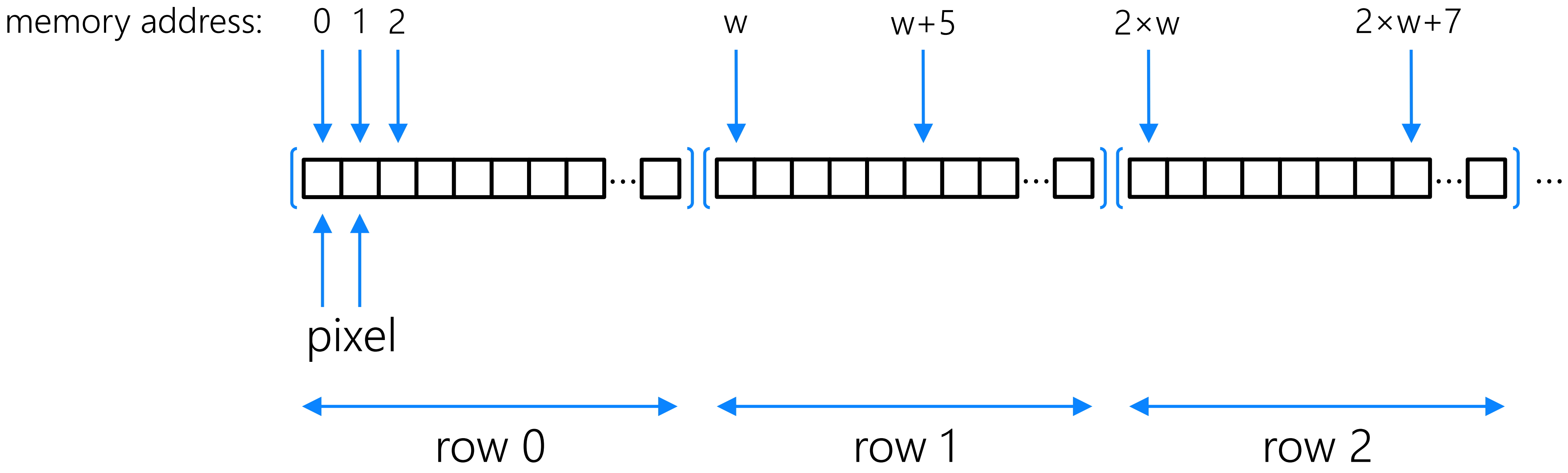
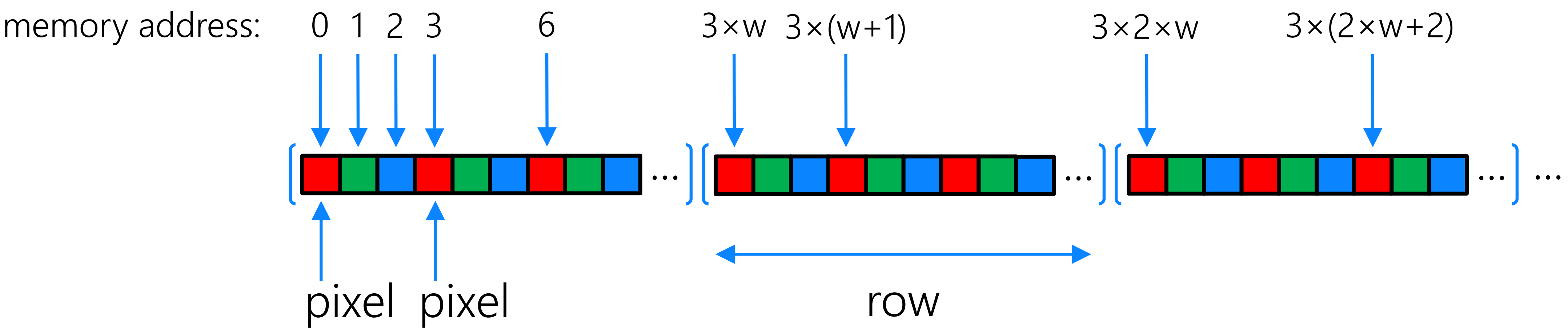


Image storage (colour)

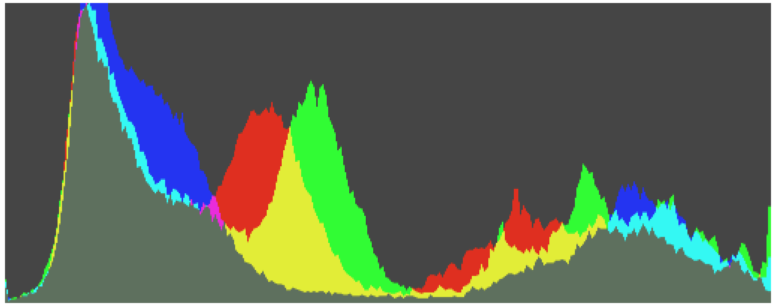
- Colour images are arranged in memory with rows of RGB pixels:



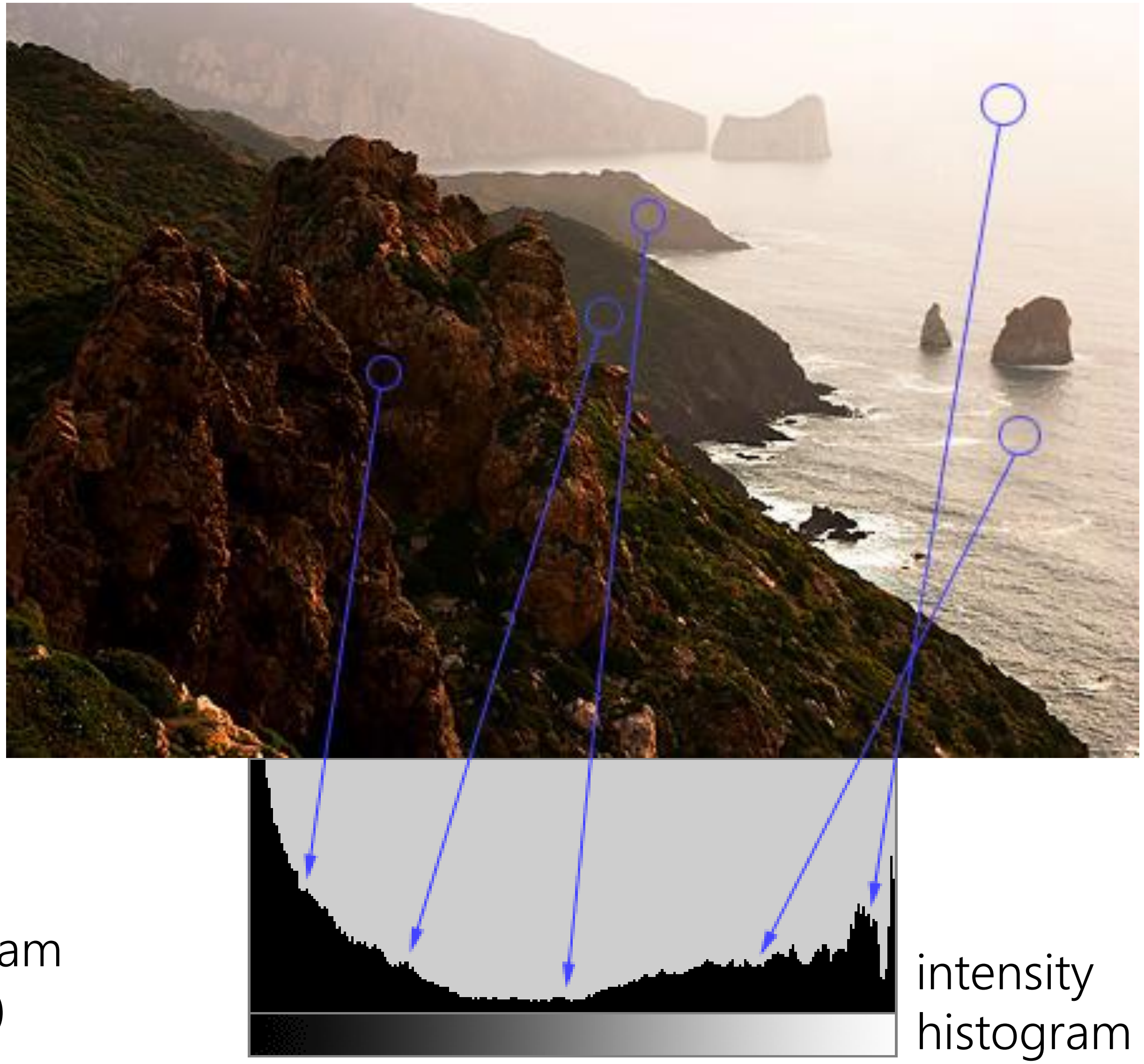
  – In general, access pixel (x, y) at memory address $3\times(y\times\text{width[in pixels]}+x)$
  – Other possibilities:
    - indexed images, 4 values per pixel (RGBA: RGB and "alpha" for opacity)

Histograms

- Summarise distribution of intensities in an image



RGB histogram (Photoshop)

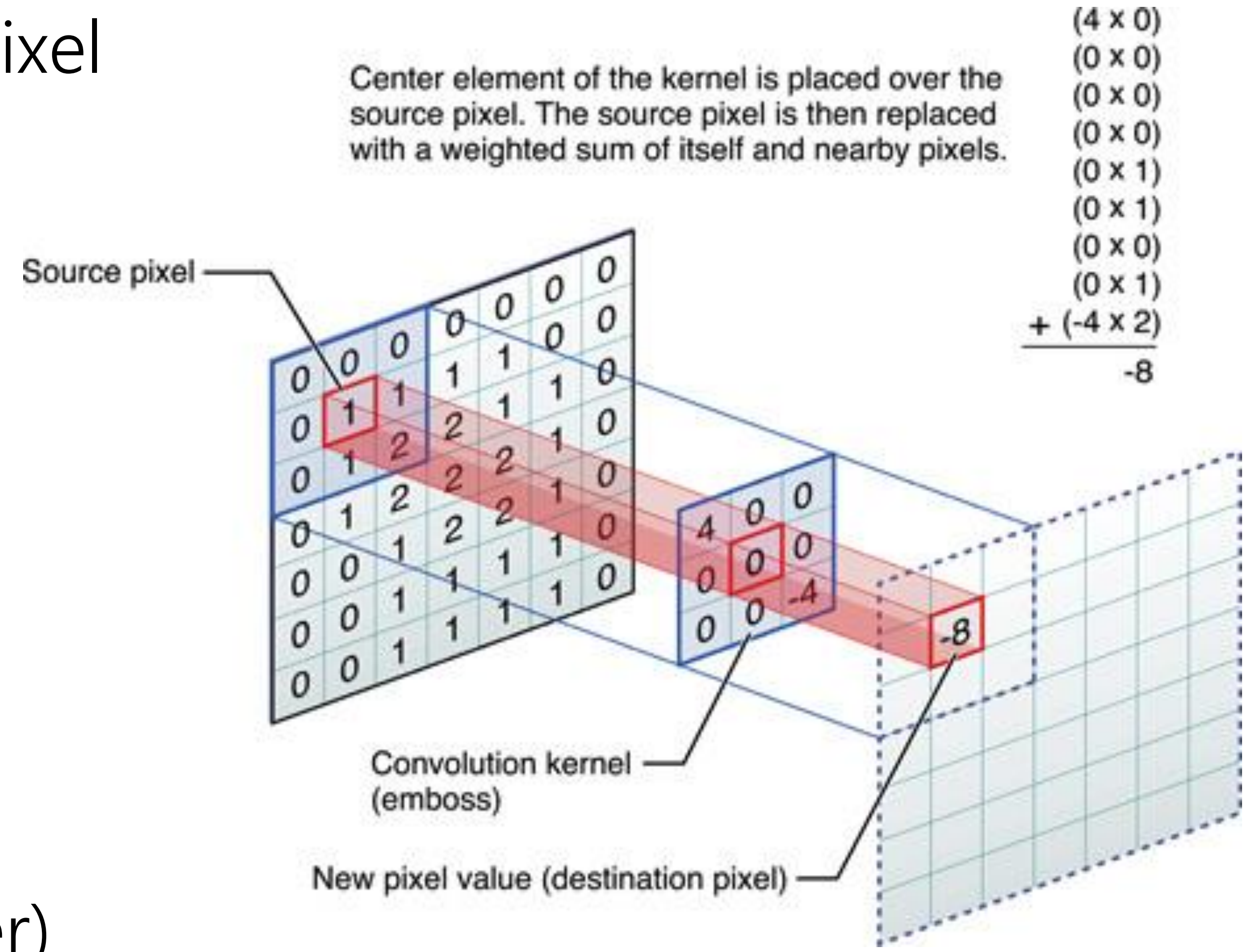


intensity histogram

Linear filtering (convolution, recap)

$$(K * I)(x, y) = \sum_i \sum_j K(i, j) I(x - i, y - j)$$

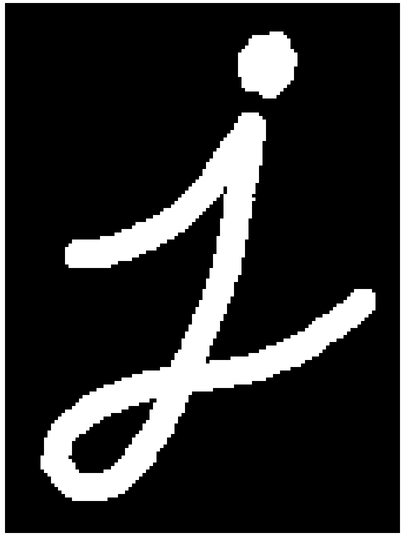
- Slide a filter over the image to compute a new value for each pixel
  - Linearly combine pixel values within a window/neighbourhood
  - Weight of each neighbouring pixel defined by a **kernel**
- **Convolution**:
  general filtering approach that supports arbitrary kernels
- (We'll look at speeding it up later)



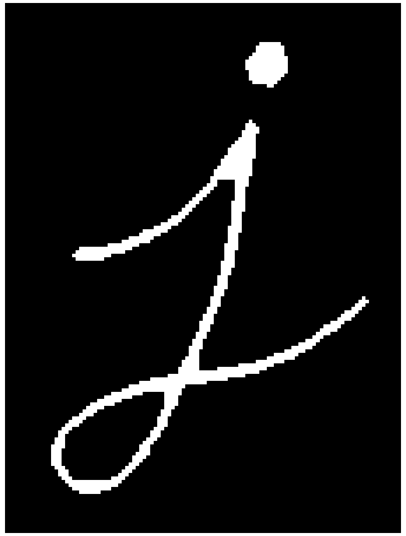
Morphological filtering (recap)

- Filtering of images using set theory
- Using a **structuring element** (set of pixel offsets), e.g. $B = \{(-2,-2), (-1,-1), (0,0), (1,1), (2,2)\}$

Input image



Erosion



Dilation

Opening

Closing

OpenCV documentation

0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
1	0	0	0	0

line

0	0	1	0	0
0	1	1	1	0
1	1	1	1	1
0	1	1	1	0
0	0	1	0	0

disc/diamond

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

square

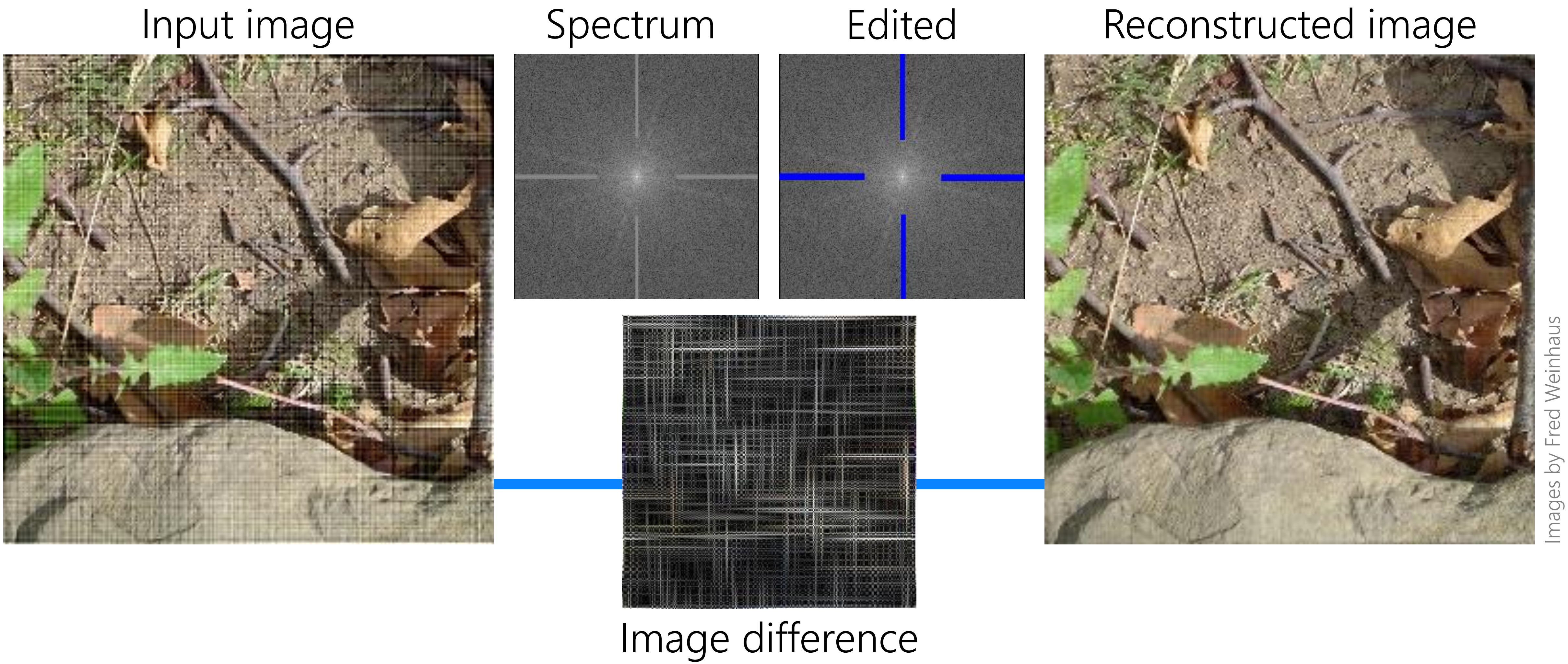
The Fourier Transform

Motivation

- Transformations are useful for analysing signals
  - Natural to analyse audio signals by decomposing into frequencies
  - Can also analyse images using frequencies in *x*- and *y*-directions
- **Fourier transform** represents signals in terms of their frequencies
- Applications:
  - Low- and high-pass filtering
  - Fast linear filtering using the convolution theorem
  - Removing structured noise
  - Image compression (JPEG)

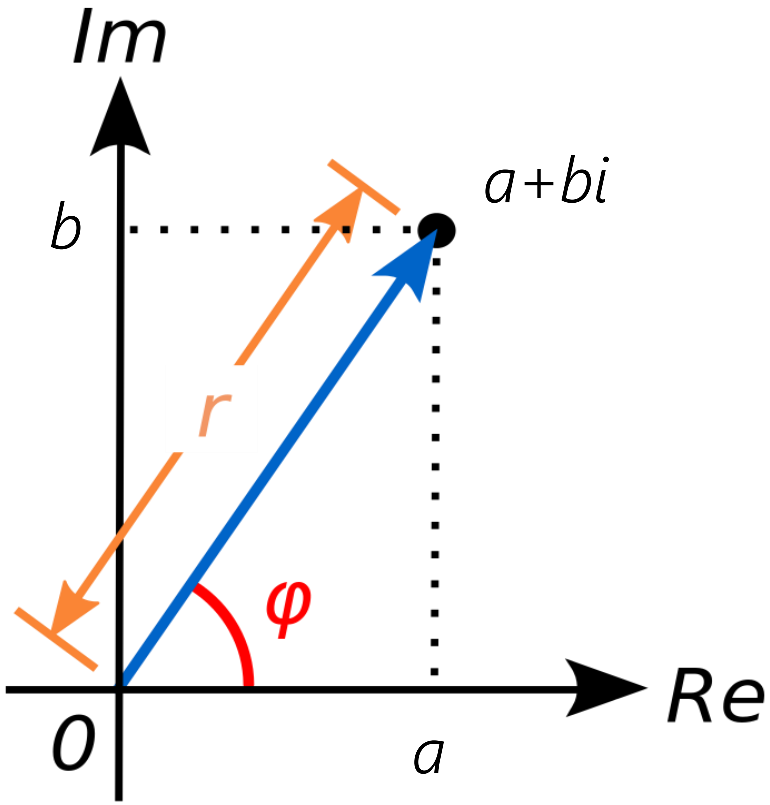
Application: structured noise removal

- Discard regions in spectrum associated with patterned noise



Complex numbers

- Extending the 1D number line to 2D complex plane: $z = a + bi \in \mathbb{C}$
  - Real part $\mathrm{Re}(z) = a \in \mathbb{R}$, imaginary part $\mathrm{Im}(z) = b \in \mathbb{R}$
  - Imaginary unit: $i^2 = -1$
- Complex conjugate: $\bar{z} = \mathrm{Re}(z) - \mathrm{Im}(z) \cdot i = a - bi$
- Polar form: $z = r \cdot \mathrm{e}^{i\varphi}$
  - Radius $r \in \mathbb{R}$, argument (angle) $\varphi \in \mathbb{R}$
  - Complex conjugate: $\bar{z} = r \cdot \mathrm{e}^{-i\varphi}$
- Magnitude: $|z| = \sqrt{z\bar{z}} = \sqrt{(a + bi)(a - bi)}$
  $= \sqrt{a^2 - b^2 i^2} = \sqrt{a^2 + b^2}$



Arithmetic with complex numbers

- Addition: $(a+bi)+(c+di)=(a+c)+(c+d)\cdot i$
- Subtraction: $(a+bi)-(c+di)=(a-c)+(c-d)\cdot i$
- Multiplication:
$$(a+bi)\cdot(c+di)=ac+adi+bci+bdi^2$$
$$=(ac-bd)+(ad+bc)\cdot i$$
  - In polar form: $(r_1 \mathrm{e}^{i\varphi_1})\cdot(r_2 \mathrm{e}^{i\varphi_2})=r_1 r_2 \mathrm{e}^{i(\varphi_1+\varphi_2)}$
- Division: $$\frac{a+bi}{c+di}=\frac{(a+bi)\cdot(c-di)}{(c+di)\cdot(c-di)}=\frac{ac-adi+cbi-bdi^2}{c^2-di^2}$$
  - expand with with complex conjugate
$$=\frac{ac+bd}{c^2+d^2}+\frac{bc-ad}{c^2+d^2}\cdot i$$

Euler's formula

$$e^{i\varphi} = \cos\varphi + i\sin\varphi$$

- Describes complex numbers on the unit circle:

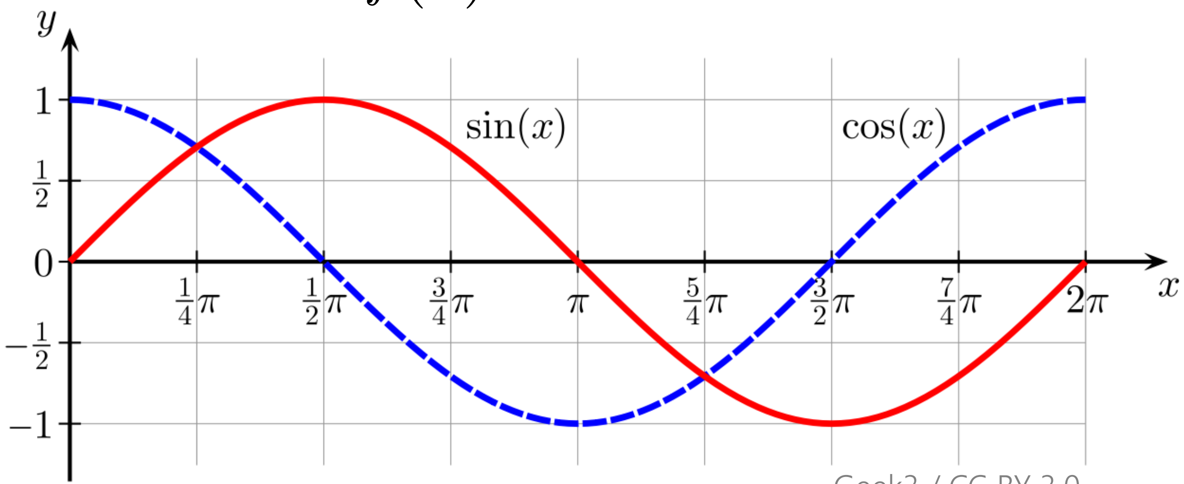
$$\begin{aligned}\left|e^{i\varphi}\right| &= \sqrt{(\cos\varphi + i\sin\varphi)(\cos\varphi - i\sin\varphi)} \\ &= \sqrt{\cos^2\varphi + \sin^2\varphi} = 1\end{aligned}$$

- Using $\sin(-\varphi) = -\sin\varphi$ and $\cos(-\varphi) = \cos\varphi$, we obtain:

$$\cos\varphi = \frac{e^{i\varphi} + e^{-i\varphi}}{2} \qquad \sin\varphi = \frac{e^{i\varphi} - e^{-i\varphi}}{2i}$$

Even and odd functions

- A function is
  - **even** if $f(-x) = f(x)$
  - **odd** if $f(-x) = -f(x)$
- Examples:
  - even: $f(x) = \cos x$
  - odd: $f(x) = \sin x$



- Properties:
  - If *f*, *g* are even, then *fg* is even
  - If *f*, *g* are odd, then *fg* is even
  - If *f* is even and *g* is odd, *fg* is odd
  - If *g* is odd, then $\int_{-a}^{a} g(x)\,dx = 0$ for $a > 0$
  - If *g* is even, then $\int_{-a}^{a} g(x)\,dx = 2\int_{0}^{a} g(x)\,dx$
- For any function *f*:

$$f(x) = f_{\mathrm{even}}(x) + f_{\mathrm{odd}}(x)$$

$$f_{\mathrm{even}}(x) = \frac{f(x) + f(-x)}{2}$$

$$f_{\mathrm{odd}}(x) = \frac{f(x) - f(-x)}{2}$$

Fourier transform

- The **Fourier transform** of a function $f : \mathbb{R} \to \mathbb{R}$ is given by:

$$F(\omega) = \mathcal{F}[f](\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) \mathrm{e}^{-i\omega x} dx$$

- The **inverse Fourier transform** restores f from F:

$$f(x) = \mathcal{F}^{-1}[F](\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(\omega) \mathrm{e}^{i\omega x} d\omega$$

- Note the similarity between the Fourier transform and its inverse.
- We use $F(\omega)$ and $\mathcal{F}[f](\omega)$ interchangeably to refer to the representation of $f(x)$ in the frequency (or Fourier) domain.

Fourier transform example: box function

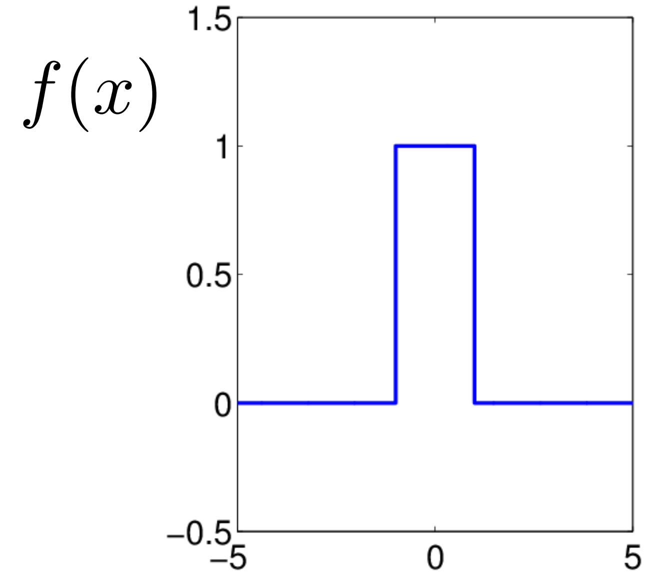
$$f(x) = \begin{cases} 1 & \text{if } -1 < x < 1 \\ 0 & \text{otherwise} \end{cases}$$

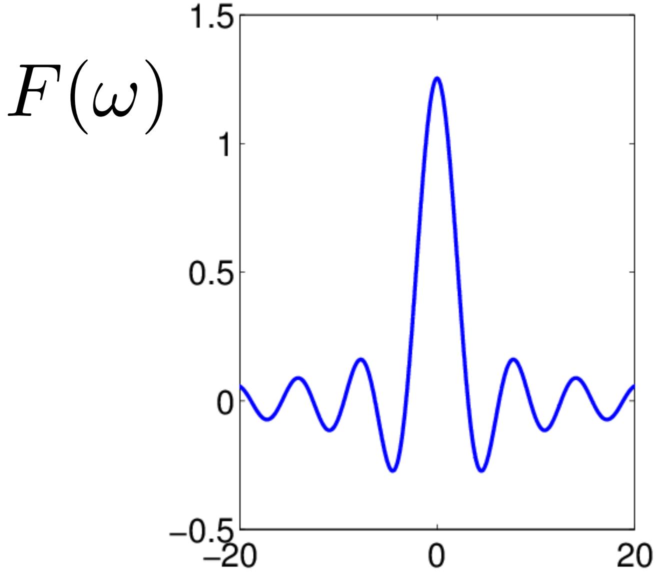
$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) \mathrm{e}^{-i\omega x} dx$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-1}^{1} \mathrm{e}^{-i\omega x} dx = \frac{1}{\sqrt{2\pi}} \left[\frac{\mathrm{e}^{-i\omega x}}{-i\omega} \right]_{-1}^{1}$$

$$= \frac{\mathrm{e}^{-i\omega} - \mathrm{e}^{i\omega}}{-i\omega\sqrt{2\pi}} = \frac{-2i\sin(\omega)}{-i\omega\sqrt{2\pi}}$$

$$= \sqrt{\frac{2}{\pi}} \frac{\sin(\omega)}{\omega} = \sqrt{\frac{2}{\pi}} \mathrm{sinc}(\omega)$$





Properties of the Fourier transform

- **Linearity:** for constants *a* and *b*, and functions *f* and *g*:

$$\mathcal{F}[af + bg](\omega) = a\mathcal{F}[f](\omega) + b\mathcal{F}[g](\omega)$$

- **Shifting:** for a constant *a*, if $g(x) = f(x - a)$ then:

$$\mathcal{F}[g](\omega) = \mathrm{e}^{-ia\omega}\mathcal{F}[f](\omega)$$

- **Modulation:** for a constant *a*, if $g(x) = \mathrm{e}^{iax} f(x - a)$ then:

$$\mathcal{F}[g](\omega) = \mathcal{F}[f](\omega - a)$$

- **Scaling:** for a constant *a*, if $g(x) = f(ax)$, then:

$$\mathcal{F}[g](\omega) = \frac{1}{|a|}\mathcal{F}[f]\left(\frac{\omega}{a}\right)$$

Properties of the Fourier transform

- If f is even and real-valued, then F is even and real-valued.
- If f is odd and real-valued, then F is odd and purely imaginary.
- Differentiation:

$$\mathcal{F}\left[\frac{df}{dx}\right](\omega) = i\omega\mathcal{F}[f](\omega)$$

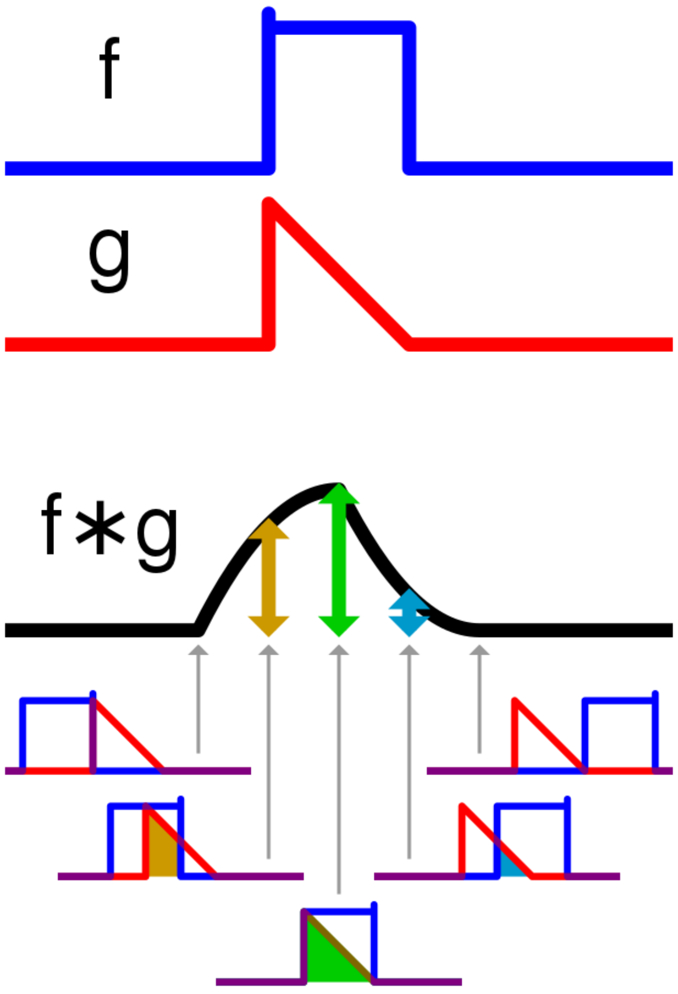
$$\mathcal{F}\left[\frac{d^n f}{dx^n}\right](\omega) = (i\omega)^n\mathcal{F}[f](\omega)$$

- Taking the derivative in the spatial domain multiplies the Fourier transform with the frequency:
  - higher frequencies are amplified, such as noise
  - Lower frequencies are attenuated

Convolution visualised

- The convolution $f * g$ of two functions *f* and *g* is given by:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x-y)g(y)\,dy$$



Convolution theorem

- The convolution $f * g$ of two functions f and g is given by:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x-y)g(y)\, dy$$

- **Convolution theorem:** $\mathcal{F}[f * g](\omega) = \sqrt{2\pi}\mathcal{F}[f](\omega) \cdot \mathcal{F}[g](\omega)$
  - Convolution becomes multiplication in the Fourier domain
  - Afterwards, transform results back to spatial domain
- As we will see, this can save a lot of time for convolutions with large kernels
- Reciprocal convolution theorem: $\mathcal{F}[f \cdot g](\omega) = \sqrt{2\pi}\mathcal{F}[f](\omega) * \mathcal{F}[g](\omega)$

Discrete Fourier transform (DFT)

- Discrete analogue to the continuous Fourier transform
  - deals with finite sampled signals, such as audio, images
  - *N* values decomposed into *N* frequency components
- **Discrete Fourier transform** (DFT):

$$F[k] = \sum_{n=0}^{N-1} f[n] \mathrm{e}^{-2\pi i n k / N} \qquad (k = 0, 1, \ldots, N-1)$$

- **Inverse discrete Fourier transform**:

$$f[n] = \frac{1}{N} \sum_{k=0}^{N-1} F[k] \mathrm{e}^{2\pi i n k / N} \qquad (n = 0, 1, \ldots, N-1)$$

Properties of the discrete Fourier transform

- Periodicity: $F[k]$ is periodic with period N since

$$F[k+N] = \sum_{n=0}^{N-1} f[n]\mathrm{e}^{-2\pi in(k+N)/N} = \sum_{n=0}^{N-1} f[n]\mathrm{e}^{-2\pi ink/N} = F[k]$$

$$\left(\mathrm{e}^{-2\pi in(k+N)/N} = \mathrm{e}^{-2\pi ink/N}\mathrm{e}^{-2\pi in} = \mathrm{e}^{-2\pi ink/N}\right)$$

- Cyclical convolution: $(f * g)[n] = \sum_{m=0}^{N-1} f[m]g[n-m]$

- **Convolution theorem:** the DFT of $f * g$ is $F[k] \cdot G[k]$

Fast Fourier transform (FFT)

- Naïve computation of DFT is expensive: $\mathcal{O}(N^2)$ complexity
- Compute N Fourier coefficients:
  - N (complex-valued) multiplications
  - N–1 (complex-valued) additions

$$F[k] = \sum_{n=0}^{N-1} f[n]\mathrm{e}^{-2\pi\mathrm{i}nk/N}$$
$$= f[0] + f[1]\mathrm{e}^{-2\pi\mathrm{i}k/N} + \cdots + f[N-1]\mathrm{e}^{-2\pi\mathrm{i}k(N-1)/N}$$

- For example, an image with 1 megapixel resolution ($N = 10^6$) would require $\mathcal{O}(N^2) = \mathcal{O}(10^{12})$ arithmetic operations
- Fast Fourier Transform (FFT) has $\mathcal{O}(N \log_2 N)$ complexity [Gauss 1805, Cooley & Tukey 1965]
  - much faster than $\mathcal{O}(N^2)$
  - e.g. $\mathcal{O}(10^6 \log_2 10^6) \approx \mathcal{O}(2 \cdot 10^7)$ for $N = 10^6$

Fast Fourier transform (FFT)

- Key insight: divide and conquer
  - Split an input of size *N* into two inputs of size N/2 each
  - Repeat until inputs of size 1 is reached
  - This is has $\mathcal{O}(\log N)$ complexity: need this many levels of nesting
- Advantages:
  - Very efficient: $\mathcal{O}(N \log_2 N)$
  - Implemented in many libraries (e.g. FFTW used by MATLAB)
- Disadvantages:
  - Requires inputs of size $N = 2^k$, so input may need padding

Discrete Fourier transform (DFT) in 2D

- Fourier techniques can be generalised to n-D functions, e.g. for 2D:
  - Discrete Fourier transform in 2D:

$$F[k,j] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[n,m] \mathrm{e}^{-2\pi i (kn/N + jm/M)}$$

$$(j = 0, 1, \ldots, M-1; k = 0, 1, \ldots, N-1)$$

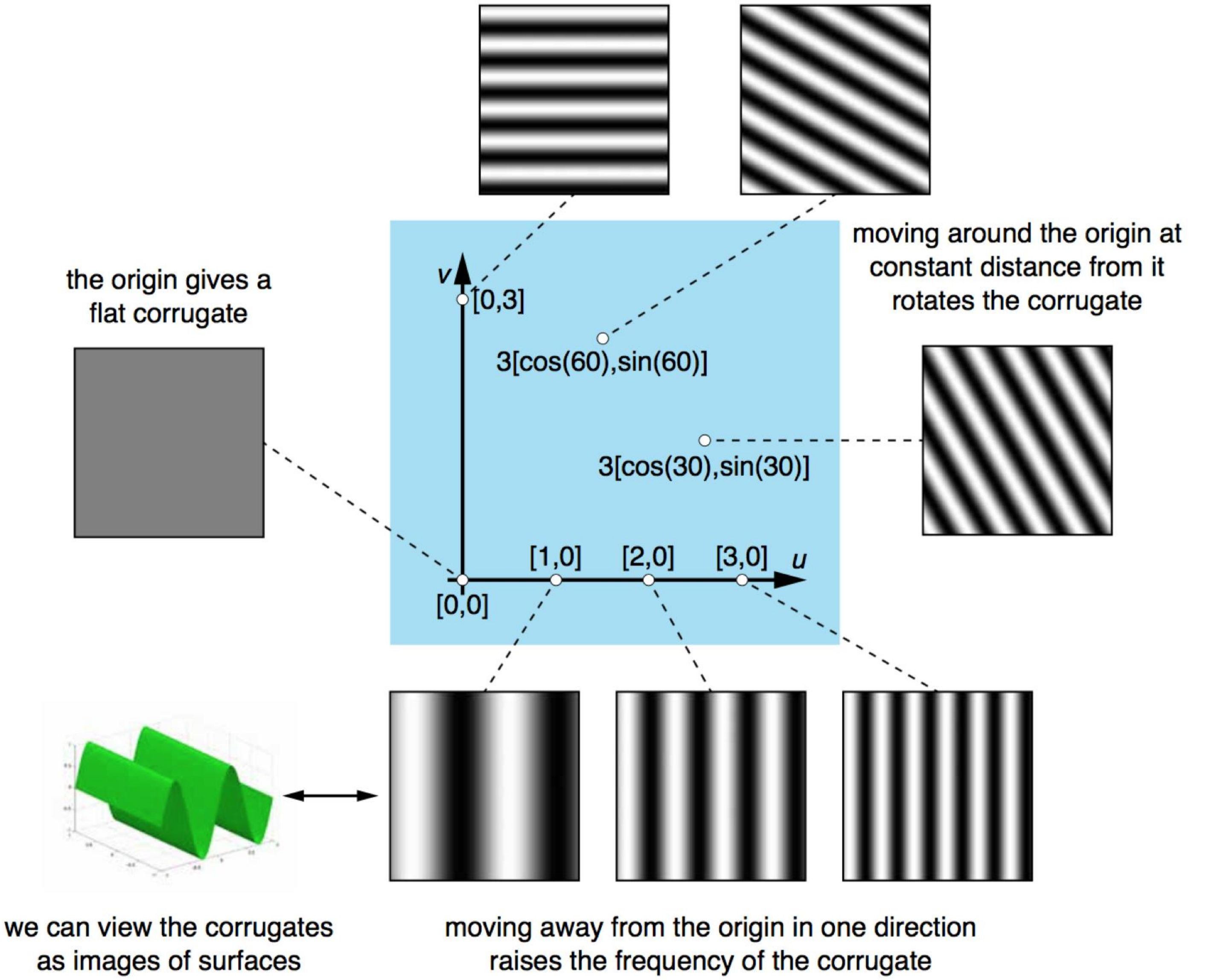
  - Discrete inverse Fourier transform in 2D:

$$f[n,m] = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F[k,j] \mathrm{e}^{2\pi i (kn/N + jm/M)}$$

$$(n = 0, 1, \ldots, N-1; m = 0, 1, \ldots, M-1)$$

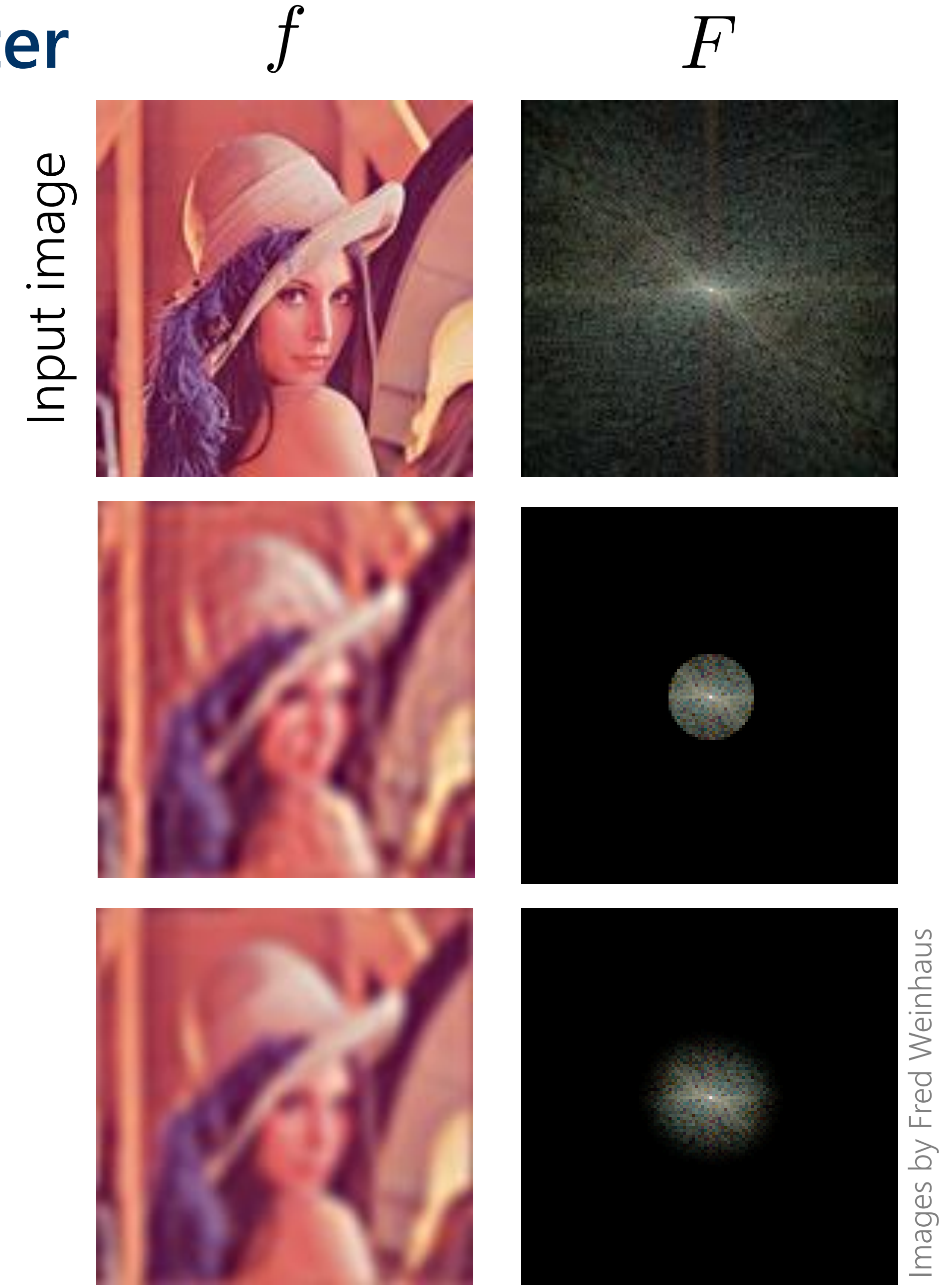
- Colour images: transform each channel independently

2D sinusoids



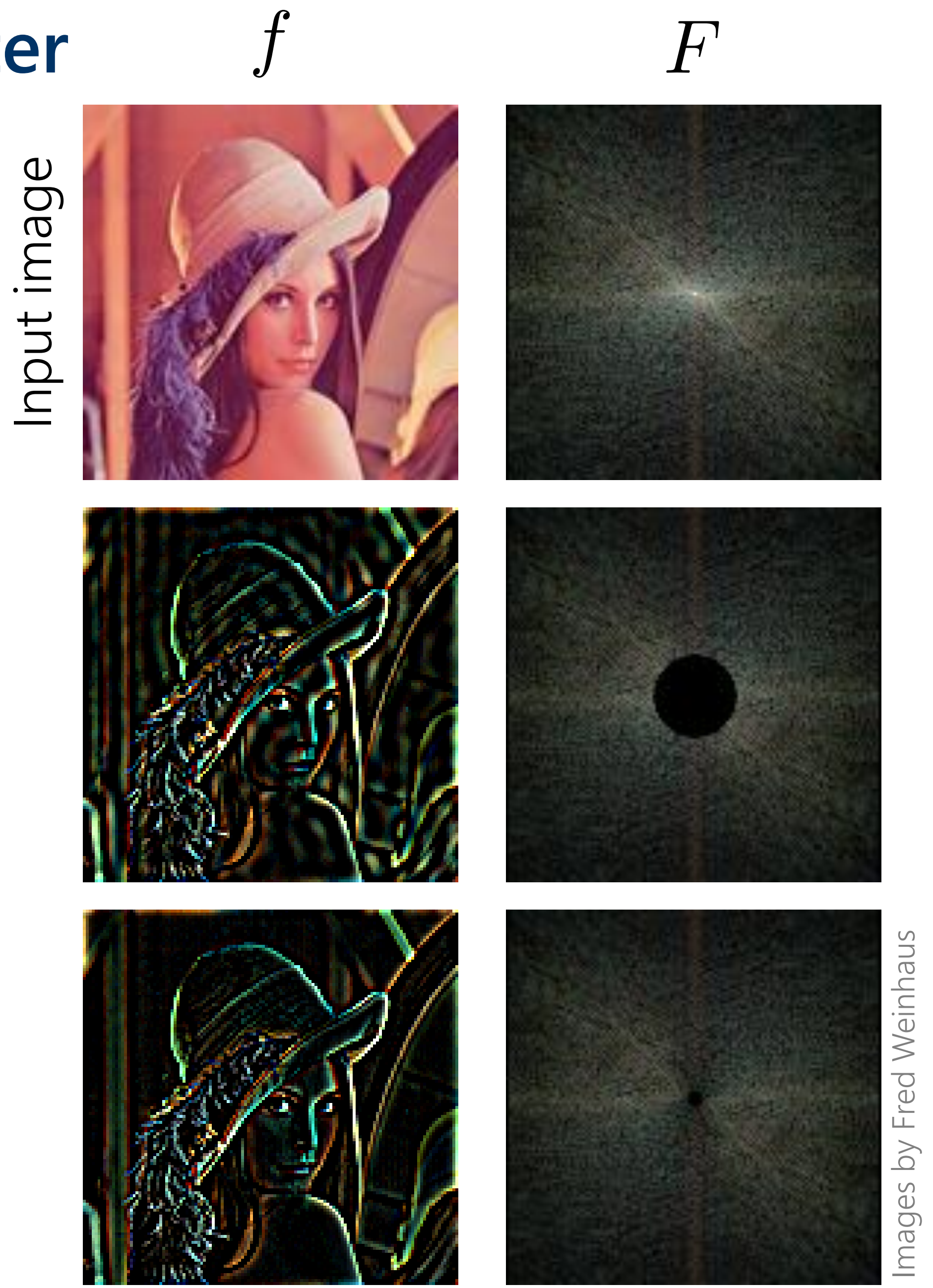
Low-pass filter

- passes signals with frequencies lower than a threshold
  - Removes fine details and hence blurs an image
- Example using hard cut-off in frequency space:
  - Ringing artefacts in image space
- Better low-pass filter: Gaussian blur
  - Removes ringing artefacts



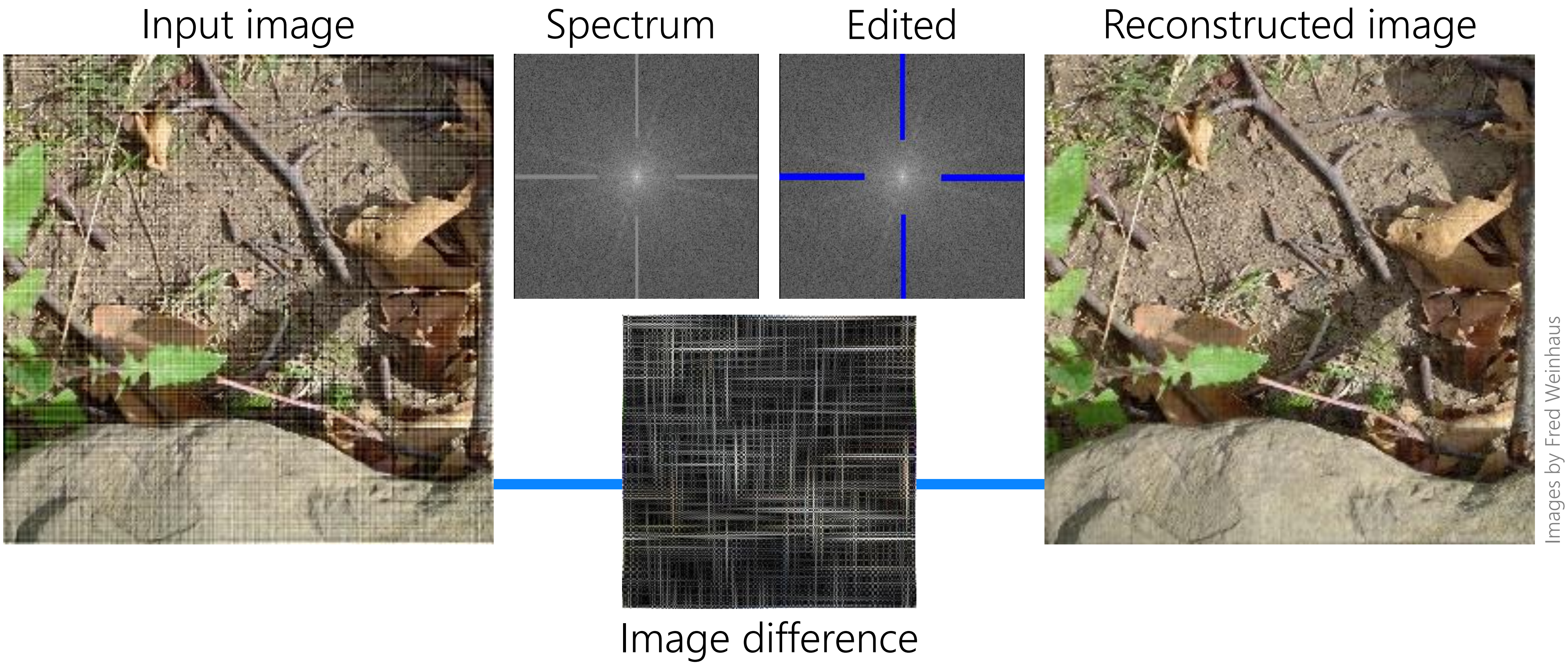
High-pass filter

- passes signals with frequencies higher than a threshold
  - Removes low frequencies (~overall shape)
  - get edge image
- Example using hard cut-off in frequency space:
  - Ringing artefacts in image space
- Better high-pass filter: 1–Gaussian blur
  - Removes ringing artefacts



Structured noise removal

- Discard regions in spectrum associated with patterned noise

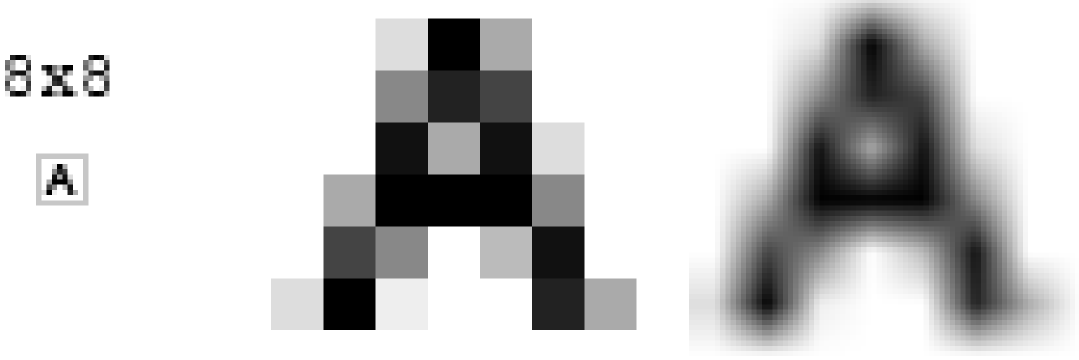


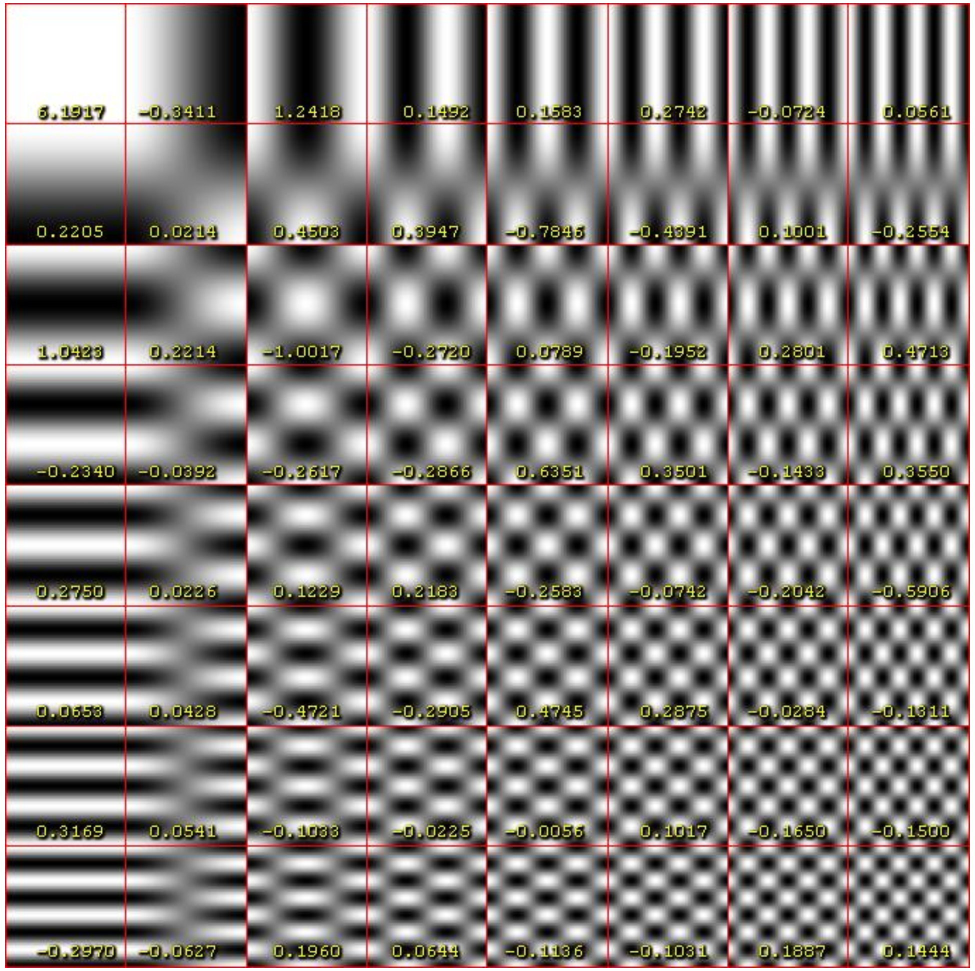
Discrete cosine transform (DCT)

- Recall that the Fourier transform of an even, real-valued signal is even and real-valued (no imaginary part)
  - Audio signals and images are real-valued
  - Can make them even by reflection around $n = 0$ or even $n = -0.5$
  - This cancels out the purely imaginary, sine-related terms
- DCTs are equivalent to DFTs of roughly twice the length
- input and/or output data shifted by half a sample in some variants
- Used for audio compression in MP3, image compression in JPEG

Application of the DCT in JPEG

- Discrete cosine transform is applied to 8×8 pixel blocks
- DCT coefficients are quantised (rounded) depending on frequency





Hanakus / Wikipedia

FYI: $$F[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[n_1, n_2] \cos\left[\frac{\pi}{N_1}\left(n_1 + \frac{1}{2}\right)k_1\right] \cos\left[\frac{\pi}{N_2}\left(n_2 + \frac{1}{2}\right)k_2\right]$$

Questions?