Image filtering

Mårten Björkman

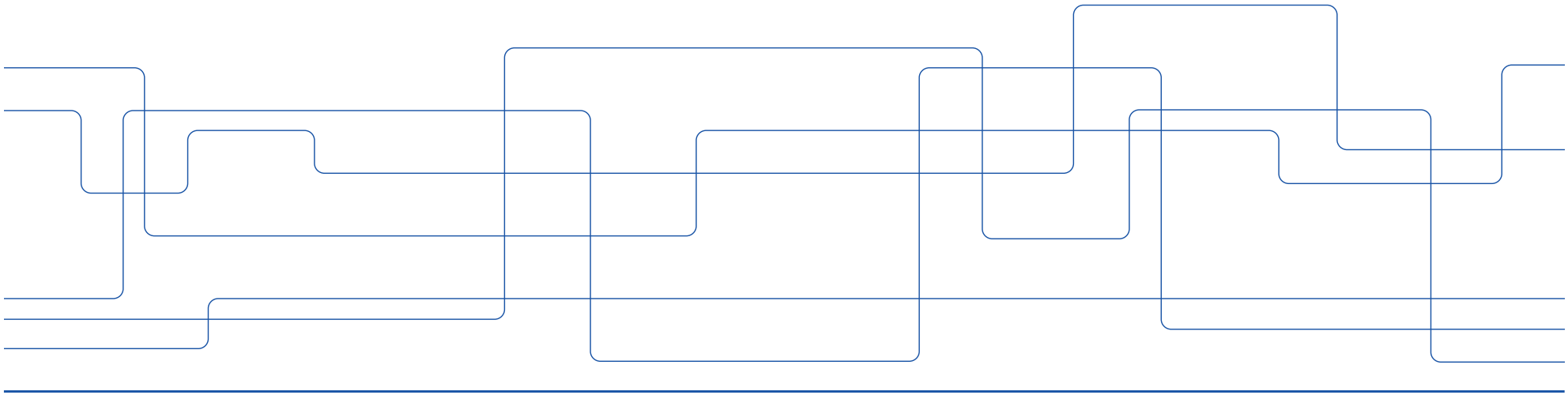
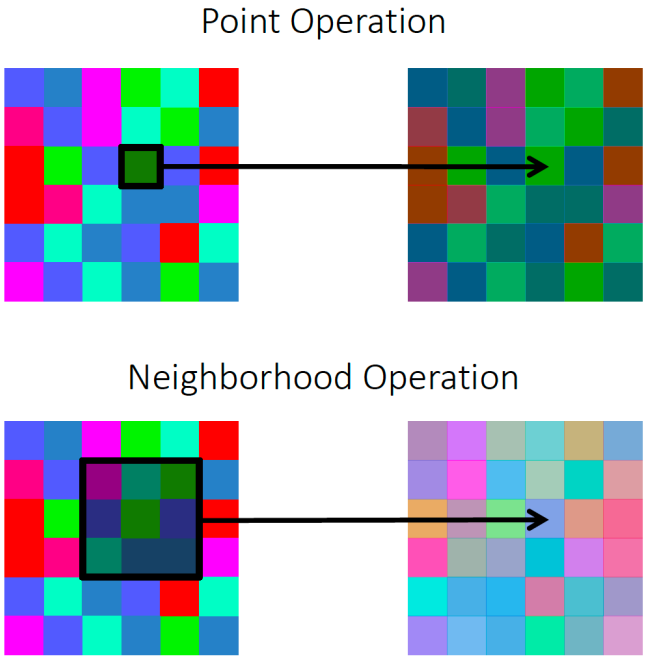


Image filtering



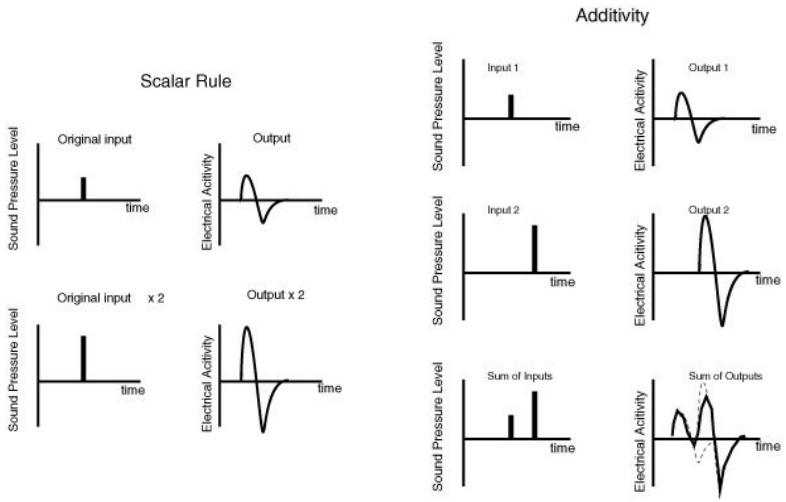
point processing

filtering

- Most spatial image filters work in local neighbourhoods. You combine a group of pixels in a neighbourhood to compute a new value for the pixel in the centre.
- Most filters are also Linear Shift-Invariant, that are easy to implement and analyse.

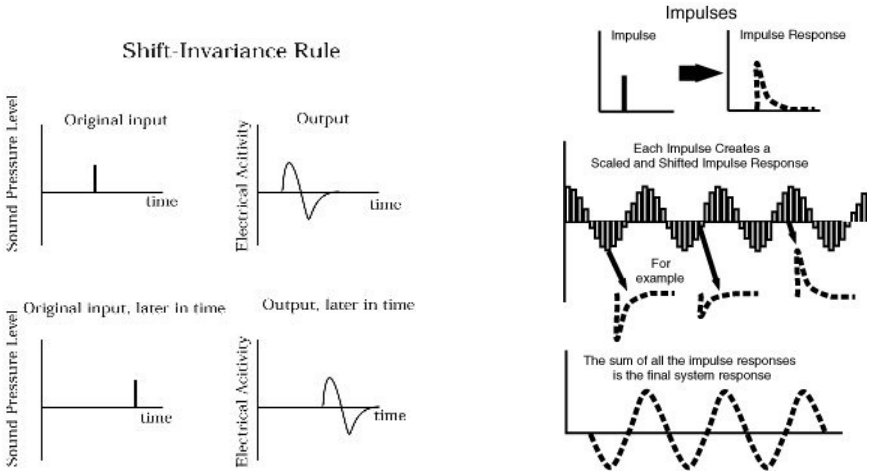
Linear operators

- Linear systems theory can be used to understand linear operators. A linear operator obeys the principle of superposition:
  - Homogeneity (scalar rule): an increase in strength of the input, increases the output/response for the same amount.
  - Additivity: if the input consists of two signals, the output/response is equal to the sum of the individual responses.



Linear shift-invariant operators

- Shift-invariance: If a system is given two impulses with a time delay, the response remains the same except for a time difference.
  - Signals can be represented as sums of impulses of different strengths (image intensities), shifted in time (image space).
- If we know how system responds to an impulse, we know how it reacts to all combinations of impulses: impulse-response function.



Linear filters in image processing

Digital linear filters modify pixel values based on some neighbourhoods. Linear means the output is a weighted combination of neighbouring pixels.

$$g(x,y) = \sum_{i,j} w(i,j) f(x+i, y+j)$$

Shift-invariance here means that the weights $w(i,j)$ are the same for all pixels.

- Benefits:
  - Linear filters are simpler and easier to understand than non-linear filters.
  - Can combine multiple linear filters applied in sequence into a single linear filter.
  - Sometimes linear filters are easier to invert, at least theoretically.

Applying a linear filter

input

f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
f_{21}	f_{22}	f_{23}	f_{24}	f_{25}
f_{31}	f_{32}	f_{33}	f_{34}	f_{35}
f_{41}	f_{42}	f_{43}	f_{44}	f_{45}
f_{51}	f_{52}	f_{53}	f_{54}	f_{55}

filter

h_{11}	h_{12}	h_{13}
h_{21}	h_{22}	h_{23}
h_{31}	h_{32}	h_{33}

output

We have a 5×5 pixel image and a 3×3 point filter that we sweep over the image.

Applying a linear filter

input

f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
f_{21}	f_{22}	f_{23}	f_{24}	f_{25}
f_{31}	f_{32}	f_{33}	f_{34}	f_{35}
f_{41}	f_{42}	f_{43}	f_{44}	f_{45}
f_{51}	f_{52}	f_{53}	f_{54}	f_{55}

filter

h_{11}	h_{12}	h_{13}
h_{21}	h_{22}	h_{23}
h_{31}	h_{32}	h_{33}

output

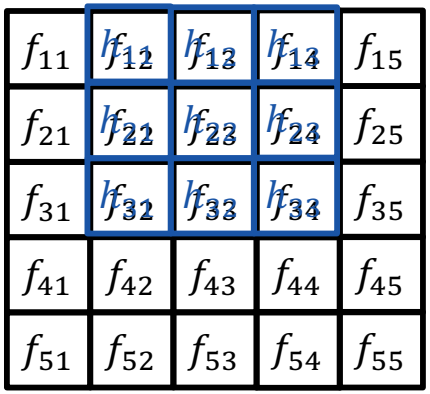
g_{11}

$$g_{11} = h_{11}f_{11} + h_{12}f_{12} + h_{13}f_{13} + h_{21}f_{21} + h_{22}f_{22} + \ldots + h_{33}f_{33}$$

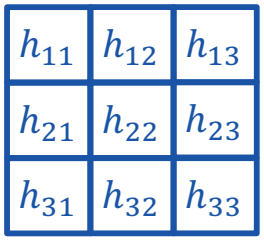
At the first position we do a point-wise multiplication and them sum up.

Applying a linear filter

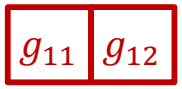
input



filter



output

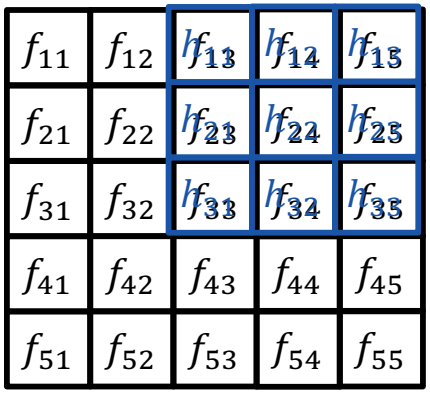


$$g_{12} = h_{11}f_{12} + h_{12}f_{13} + h_{13}f_{14} + h_{21}f_{22} + h_{22}f_{23} + \ldots + h_{33}f_{34}$$

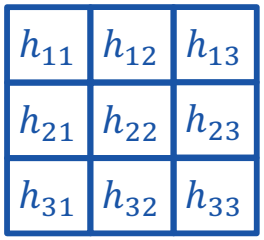
At the second position we do a point-wise multiplication and them sum up.

Applying a linear filter

input



filter



output

$$g_{13} = h_{11}f_{13} + h_{12}f_{14} + h_{13}f_{15} + h_{21}f_{23} + h_{22}f_{24} + \ldots + h_{33}f_{35}$$

At the third position we do a point-wise multiplication and them sum up.

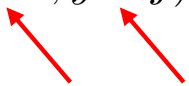
Cross-correlation vs Convolution

So far we have used a cross-correlation

$$g(x,y) = \sum_{i,j} h(i,j) f(x+i, y+j) \qquad g = h \otimes \mathrm{f}$$

But usually we do convolutions instead

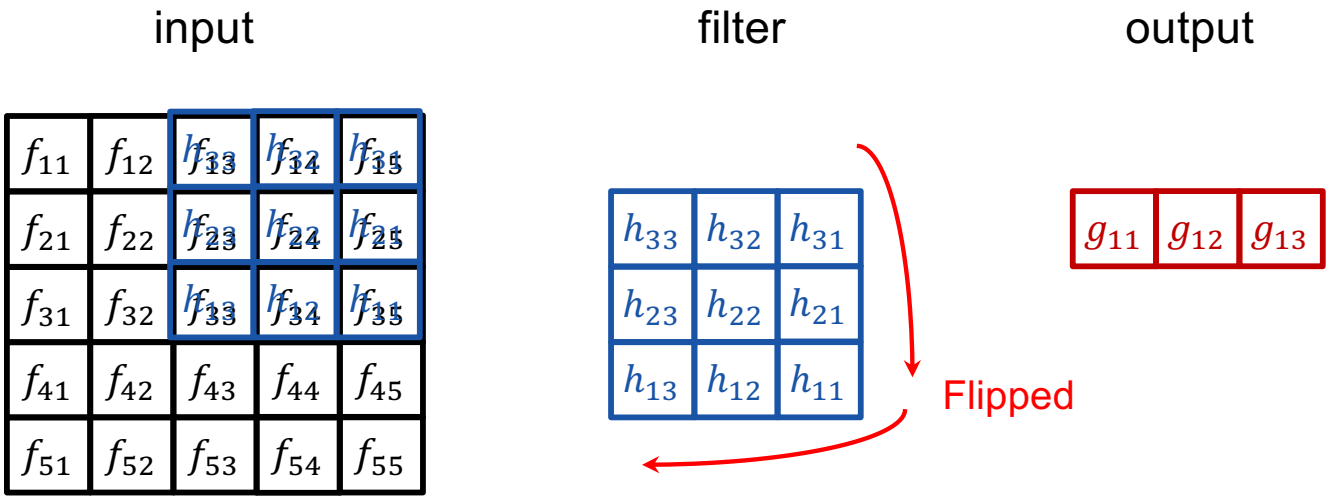
$$g(x,y) = \sum_{i,j} h(i,j) f(x-i, y-j) \qquad g = h * \mathrm{f}$$



Reasons:

- Linear system theory tells us that any linear shift-invariant (LSI) operator can be written as a convolution.
- Convolutions are commutative, e.g. $g = h * f = f * h$, so it doesn't matter in which order you apply the filters.

Applying a linear filter with a convolution



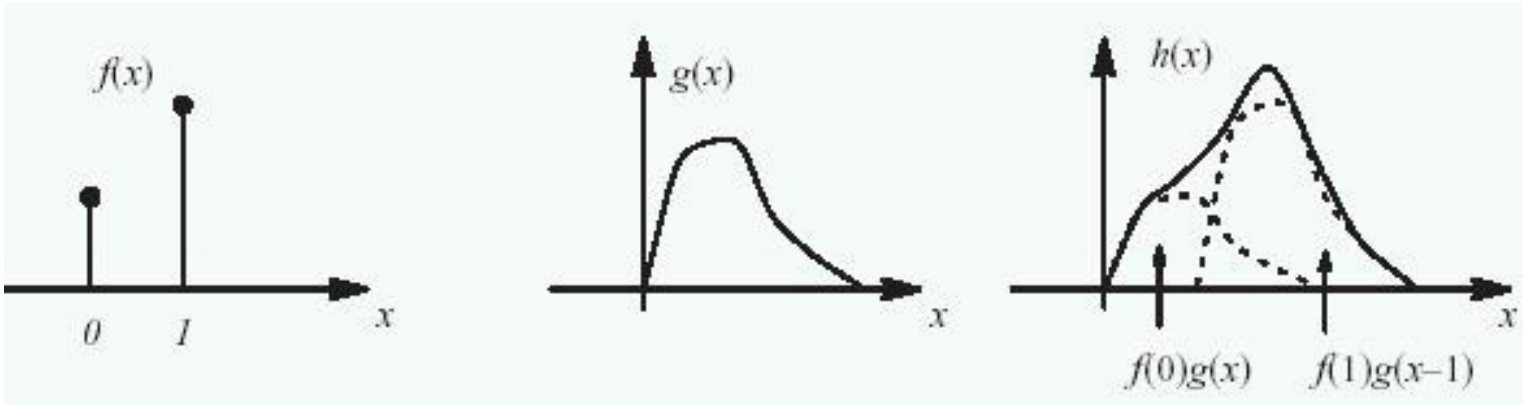
$$g_{13} = h_{33}f_{13} + h_{32}f_{14} + h_{31}f_{15} + h_{23}f_{23} + h_{22}f_{24} + \ldots + h_{11}f_{35}$$

Essentially the same, but with filter kernel flipped x-wise and y-wise.

Understanding convolutions

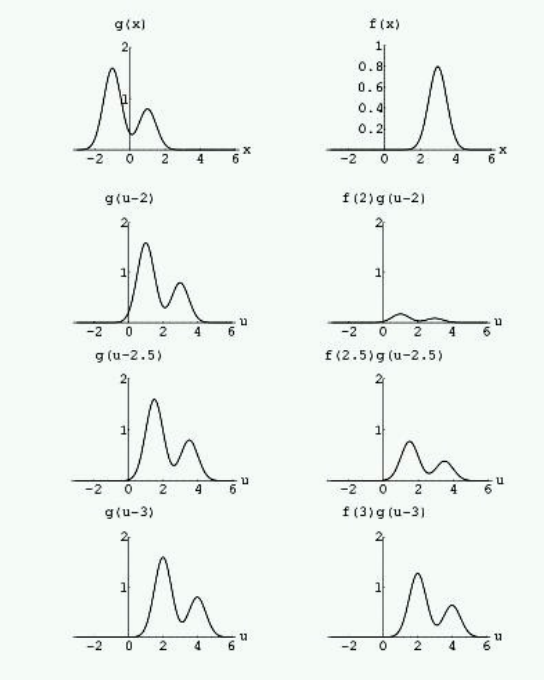
- Mathematically, a convolution is defined as the integral over space of one function at α, times another function at $x - \alpha$.

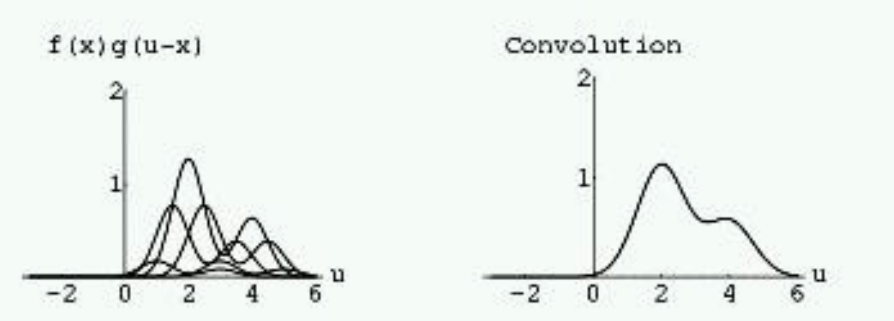
$$\mathrm{f}(x) * \mathrm{g}(x) = \int_{\alpha} \mathrm{f}(\alpha)\mathrm{g}(x-\alpha)d\alpha = \mathrm{g}(x) * \mathrm{f}(x) = \int_{\alpha} \mathrm{g}(\alpha)\mathrm{f}(x-\alpha)d\alpha$$



Understanding convolutions

- Way of considering convolution: weighted sum of shifted copies of one function, with weights given by the function value of the second function.





The filter kernel

$$g(x,y) = \sum_{i,j} h(i,j) f(x-i, y-j)$$

h_{11}	h_{12}	h_{13}
h_{21}	h_{22}	h_{23}
h_{31}	h_{32}	h_{33}

- In the discrete case, the filter is represented by an array of weights.
- It goes by many names:
  - Impulse response
  - Point spread function
  - Filter kernel
  - Filter mask
  - Template

Convolution: 1D example

If

$F_1 = [1, 2, 3, 4, 5]$

$F_2 = [1, 2, 1, 2, 1]$

$G_1 = [-1, 2, -1]$

$G_2 = [1, 2, 3]$

then

$F_1 * G_1 = [-1, 0, 0, 0, 0, 6, -5]$

$F_2 * G_1 = [-1, 0, 2, -2, 2, 0, -1]$

$F_1 * G_2 = [1, 4, 10, 16, 22, 22, 15]$

$F_2 * G_2 = [1, 4, 8, 10, 8, 8, 3]$

- Note1: outside the windows, values are usually assumed to be zero.
- Note2: normally you assume $x = 0$ at centre of filter kernel.

Convolution: 1D example

$F_1 = [1, 2, 3, 4, 5]$

$G_2 = [1, 2, 3]$

			1	2	3	4	5
$*$					1	2	3
			3	6	9	12	15
		2	4	6	8	10	
$+$	1	2	3	4	5		
	1	4	10	16	22	22	15

An easier way of doing it! Almost like regular multiplication.

Convolutions in Matlab and Python

In `Matlab` (and `SciPy.signal`) there are essentially two functions that can be used for image filtering.

- `conv2 (convolve2d)` – a proper convolution according to the theory.

$$g(x) = \sum_i h(i)\ f(x-i)$$

Use it when you want to stick as close as possible to the theory and exploit known mathematical properties of convolutions.

- `filter2 (correlate2d)` – does not flip the kernel before applying it.

$$g(x) = \sum_i h(i)\ f(x+i)$$

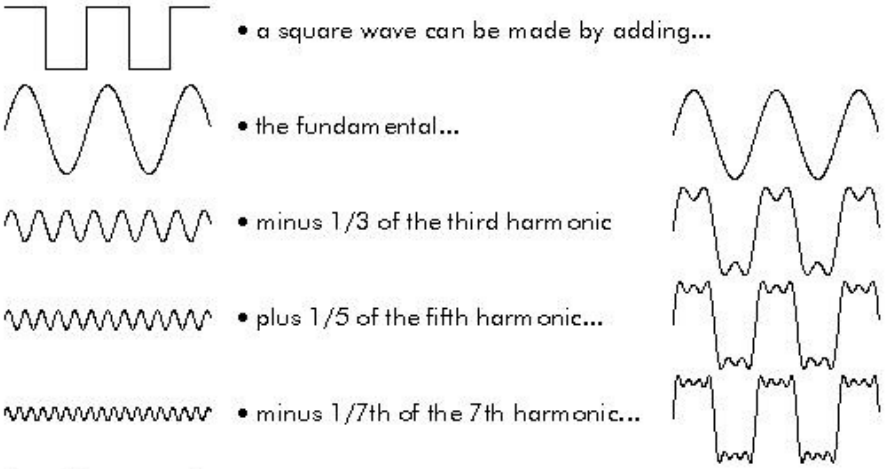
Use it when you want to match a known shape to the image data to see where in the image you have the best correlation.

Signal decomposition

- We can think of an image as a weighted sum of pixels, but we can think of it as a weighted sum of something else.
- In 1807 Jean Baptiste Fourier showed that any periodic signal can be represented by a sum of sine waves with appropriate amplitude, frequency and phase.

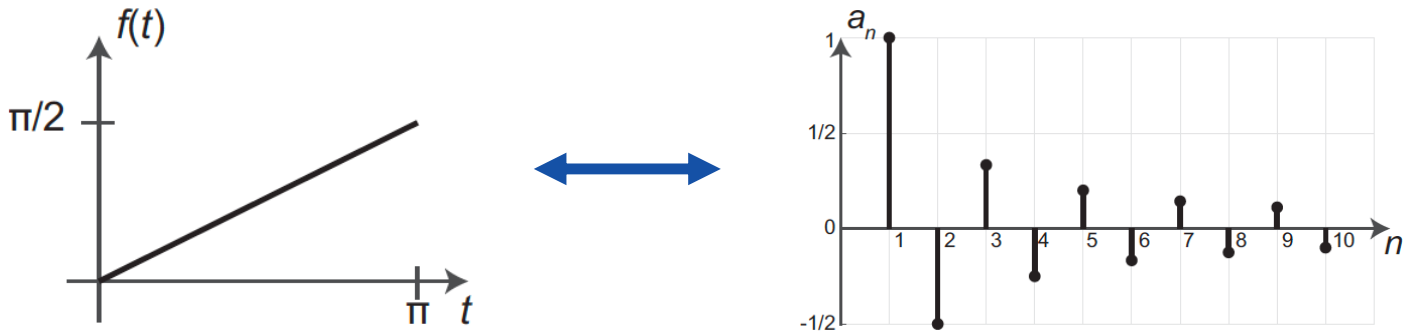
Amplitude Frequency Phase

$$A\ sin(\omega x + \alpha)$$



Fourier series

$$\frac{1}{2}t = sin(t) - \frac{1}{2}sin(2t) + \frac{1}{3}sin(3t) - \frac{1}{4}sin(4t) + ...$$



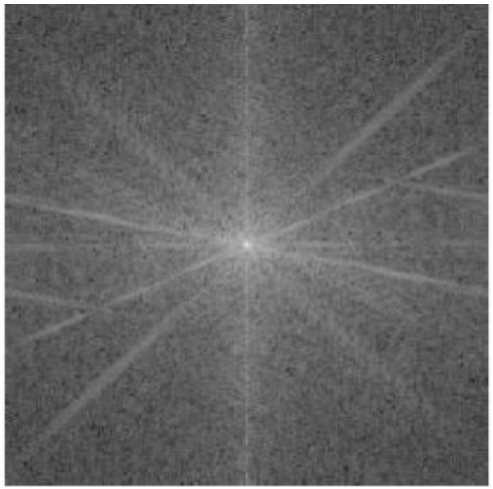
spatial domain

frequency domain

- By writing the signal (image) as a sum of sinusoids, you get an alternative frequency space representation of the same signal.

Fourier Transform

- The Fourier transform is an operation that calculates the frequency, amplitude and phase of each sine wave needed to make up any given signal.
- In the Fourier space, each point represents a particular frequency contained in the original spatial domain image.
- The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, and image compression.



Fourier Transform in 2D

$$\mathcal{F}(\mathrm{f}(x)) = \int_{x\in\mathbb{R}^2} \mathrm{f}(x) e^{-i\omega^T x} dx = \hat{\mathrm{f}}(\omega)$$

$$\mathcal{F}^{-1}(\hat{\mathrm{f}}(\omega)) = \frac{1}{(2\pi)^2} \int_{\omega\in\mathbb{R}^2} \hat{\mathrm{f}}(\omega) e^{i\omega^T x} d\omega = f(x)$$

$$e^{i\omega^T x} = \cos\omega^T x + i\sin\omega^T x, \quad \omega^T x = \omega_1 x_1 + \omega_2 x_2$$

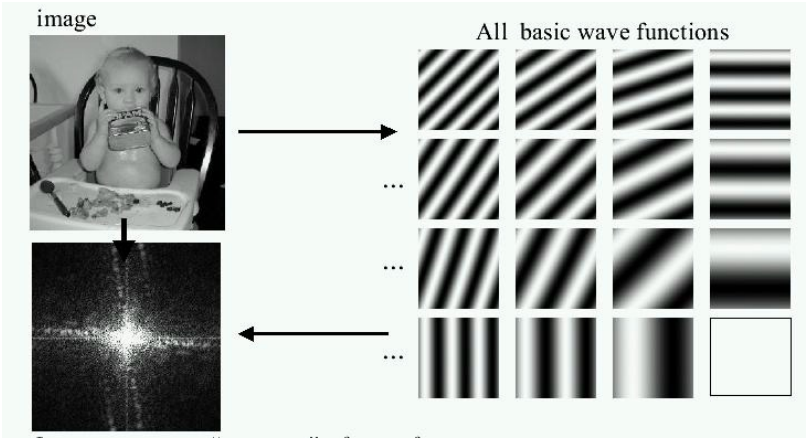
Terminology:

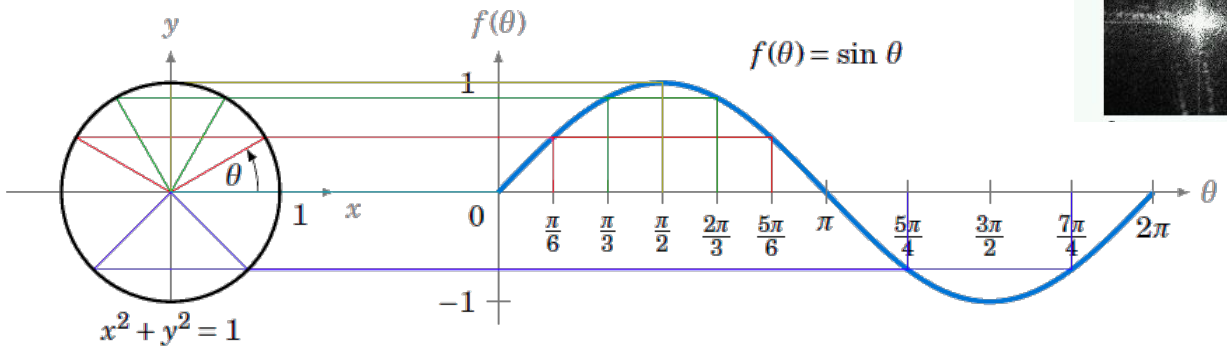
- Frequency spectrum: $\hat{f}(\omega) = Re[\hat{f}(\omega)] + i\, Im[\hat{f}(\omega)] = \left|\hat{f}(\omega)\right| e^{-i\phi(\omega)}$
- Angular frequency: $\omega = \begin{pmatrix}\omega_1\\ \omega_2\end{pmatrix}$
- Frequency: $f = \frac{\omega}{2\pi}$
- Wavelength: $\lambda = \frac{2\pi}{||\omega||} = \frac{2\pi}{\sqrt{{\omega_1}^2+{\omega_2}^2}}$

Fourier Transform in 2D

$$Re[\hat{\mathrm{f}}(\omega)] = \int_x \mathrm{f}(x)\cos(\omega^T x)dx$$

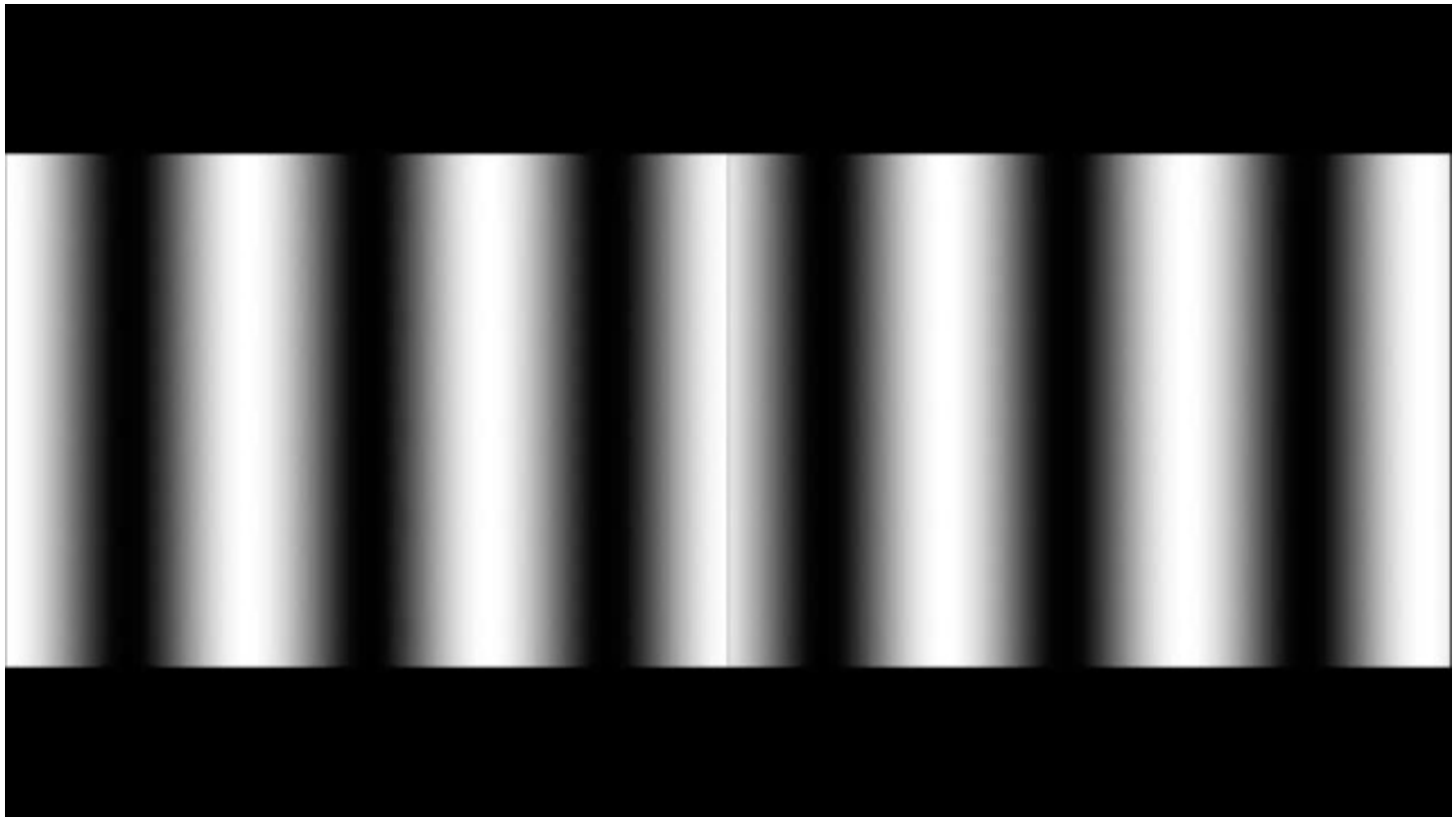
$$Im[\hat{\mathrm{f}}(\omega)] = -\int_x \mathrm{f}(x)\sin(\omega^T x)dx$$





- The complex exponential function $e^{-i\omega^T x}$ can be seen as rotating around the unit circle.
- You correlate $f(x)$ twice, first with $cos(\omega^T w)$ and then with $sin(\omega^T w)$.

Image decomposition



Gradually reconstruct image by summing up waveforms in order of decreasing magnitudes

Discrete Fourier Transform (DFT)

$$\hat{\mathrm{f}}(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} \mathrm{f}(x,y) e^{-2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)}$$

$$\mathrm{f}(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} \hat{\mathrm{f}}(u,v) e^{+2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)}$$

Terminology:

- Frequency spectrum: $\hat{f}(u,v) = Re[\hat{f}(u,v)] + i\, Im[\hat{f}(u,v)] = \left|\hat{f}(u,v)\right| e^{-i\phi(u,v)}$
- Power spectrum: $P(u,v) = |\hat{f}(u,v)|^2 = Re[\hat{f}(u,v)]^2 + Im[\hat{f}(u,v)]^2$

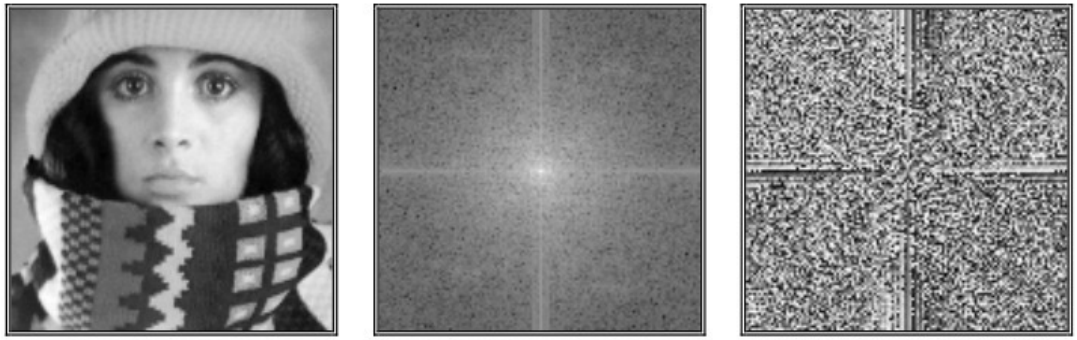
The magnitude $\left|\hat{f}(u,v)\right|$ is simply the peak value, and the phase $\phi(u,v)$ determines where the origin is, or where the sinusoid starts.

Magnitude and Phase

- The Fourier coefficients $\hat{f}(u,v)$ are complex numbers, but it is not obvious what the real and imaginary parts represent.
- Another way to represent the data is with phase and magnitude.

Magnitude: $$|\hat{\mathrm{f}}(u,v)| = \sqrt{Re^2[\hat{f}(u,v)] + Im^2[\hat{f}(u,v)]}$$

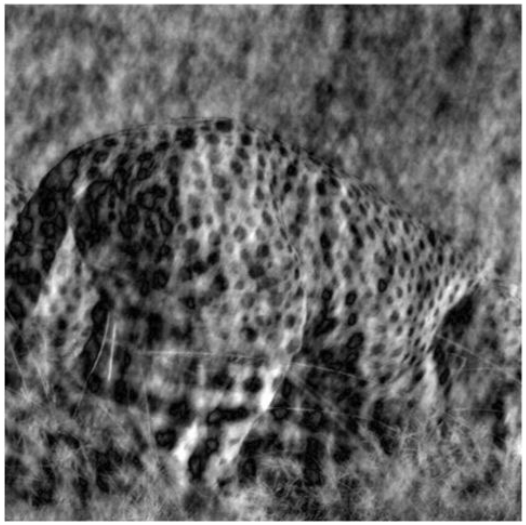
Phase: $$\phi(u,v) = \tan^{-1}\frac{Im[\hat{f}(u,v)]}{Re[\hat{f}(u,v)]}$$



Magnitude and Phase

- Phase defines how waveforms are shifted along its direction.
  - Where edges will end up in the image (most important).
- Magnitude defines how large the waveforms are.
  - What grey-levels are on either side of edges (less important).

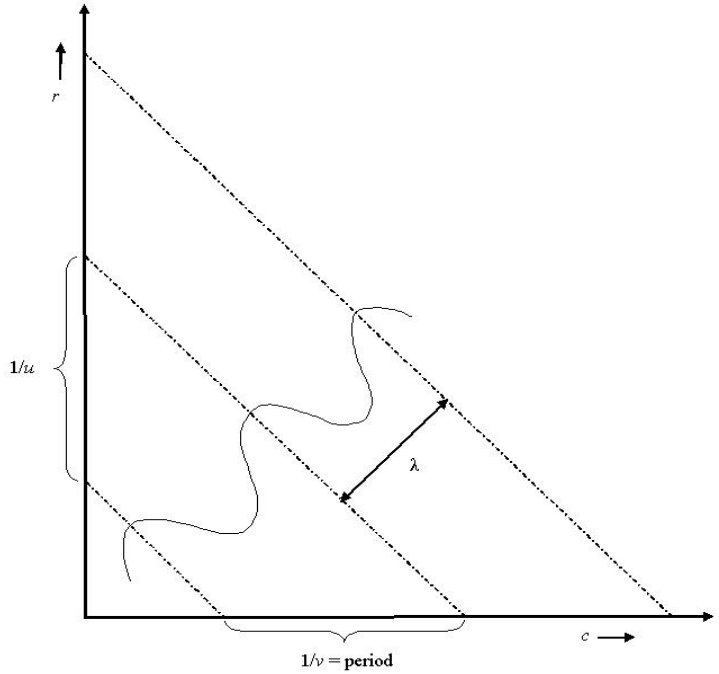
Phase of zebra - magnitude of tiger



Phase of tiger - magnitude of zebra

Wavelength

- The wavelength of the sinusoid is: $\lambda = \frac{1}{\sqrt{u^2+v^2}}$, where (u, v) are the frequencies along (x, y) and the periods are $1/u$ and $1/v$.



Change of basis functions

- An image can be viewed as a spatial array of grey-level values, but can also be thought of as a spatially varying function.

- It is possible to decompose the image into a set of orthogonal basis functions.
- When basis functions are combined (linearly) the original function will be reconstructed.
  - Spatial domain: basis consists of Dirac pulses, one per pixel.
  - Fourier domain: basis consists of complex exponential functions.

- The Fourier transform is “just” a change of basis functions!

Change of basis functions

- Assume you have a vector

$$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} = 1 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + 2 \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + 3 \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + 4 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

- This may be expressed with another basis (e.g. Haar wavelet).

$$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} = 2.5 \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} - 1 \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} - 0.5 \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix} - 0.5 \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \end{pmatrix}$$

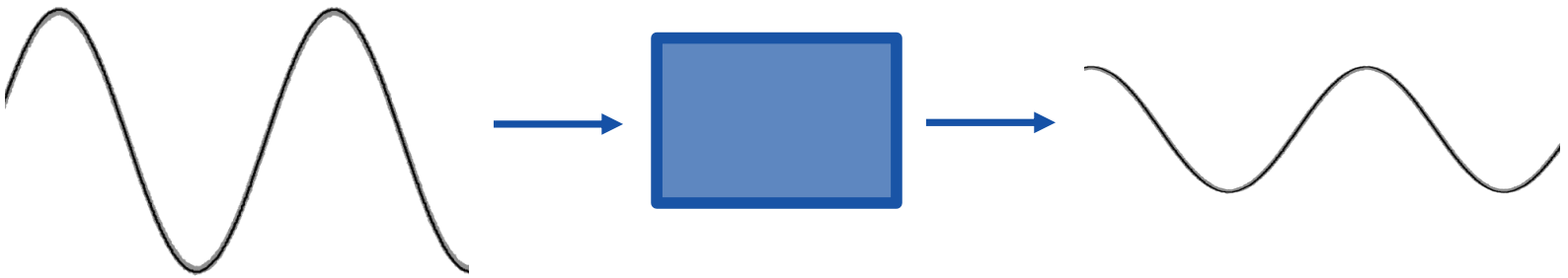
- The only condition is that the basis vectors are orthogonal.

Why Fourier transforms?

- Complex exponential functions are eigenfunctions of convolutions!

$$e^{i\omega t} \Longrightarrow A(\omega)e^{i\omega t}$$

- Note: $A(\omega)$ is complex (change in magnitude and phase).
- We can understand what the filter does to the different frequencies of the image.



The Convolution Theorem

- Convolution in the spatial domain is the same as multiplication in the Fourier (frequency) domain.

$$\mathcal{F}(\mathrm{h} * \mathrm{f}) = \mathcal{F}(\mathrm{h})\mathcal{F}(\mathrm{f})$$

$$\mathrm{f} \rightarrow \boxed{*h} \rightarrow \mathrm{g}{=}\mathrm{h}{*}\mathrm{f} \qquad \hat{\mathrm{f}} \rightarrow \boxed{\hat{\mathrm{h}}} \rightarrow \hat{\mathrm{g}} = \hat{\mathrm{h}}\,\hat{\mathrm{f}}$$

- (vice versa) Multiplication in the spatial domain is the same as convolution in the Fourier (frequency) domain.

$$\mathcal{F}(\mathrm{hf}) = \mathcal{F}(\mathrm{h}) * \mathcal{F}(\mathrm{f})$$

- Usage:
  - For analysis and understanding of linear shift-invariant filters.
  - Some filters are more easily represented in the Fourier domain.
  - Implementation: when the size of the filter is too large, it is more effective to use multiplication in the Fourier domain.

Transfer functions

- A linear, shift-invariant filter $h(x, y)$ is completely specified by it's response to an impulse, which is called the impulse response.
- Convolution with an impulse function

$$h(x,y) * \delta(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x_0,y_0)\delta(x-x_0,y-y_0)dx_0dy_0 = h(x,y)$$

results in a "copy" of $h(x, y)$ to the location of the impulse.

- The transfer function $H(u, v)$ is the Fourier transform of the impulse response.
- Using the convolution theorem to describe the effects of the system:

$$g(x,y) = h(x,y) * f(x,y)$$

$$G(u,v) = H(u,v)F(u,v)$$

Spatial separability

- Assume a filter can be written as

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

$$\mathrm{h}(x, y) = \mathrm{h}_1(x)\mathrm{h}_2(y)$$

$$(\mathrm{h}{*}\mathrm{f})(x, y) = \sum_{i=0}^{m}\sum_{j=0}^{m} \mathrm{h}(i, j)\mathrm{f}(x - i, y - j) = \sum_{i=0}^{m} \mathrm{h}_1(i)(\sum_{j=0}^{m} \mathrm{h}_2(j)\mathrm{f}(x - i, y - j))$$

- If convolution mask $h(x, y)$ can be separated as above, 2D convolution can be performed as a series of 1D convolutions.
- Discrete case: If the mask is m^2 in size → $2m$ operations per pixel, instead of m^2 operations per pixel.

Separability in the Fourier domain

Fourier transforms are always separable

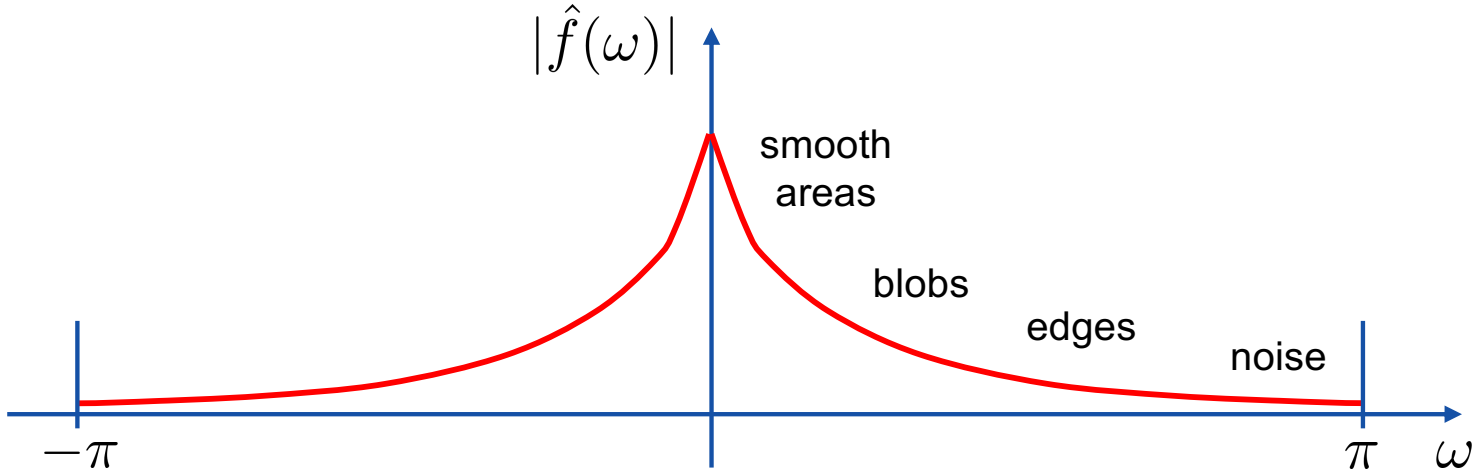
$$\hat{\mathrm{f}}(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} \mathrm{f}(x,y)e^{-2\pi i(\frac{xu}{M}+\frac{yv}{N})} = \sum_{x=0}^{M-1}(\sum_{y=0}^{N-1} \mathrm{f}(x,y)e^{-2\pi i\frac{xu}{M}})e^{-2\pi i\frac{yv}{N}}$$

- A Fourier transform in 2D can always be performed as a series of two 1D Fourier transforms.
- Implication: For large non-separable spatial filters, it is faster to
  1. Go to the frequency domain (Fourier transform)
  2. Apply the filter as a point-wise multiplication
  3. Go back to the spatial domain (Inverse Fourier transform)

Applications

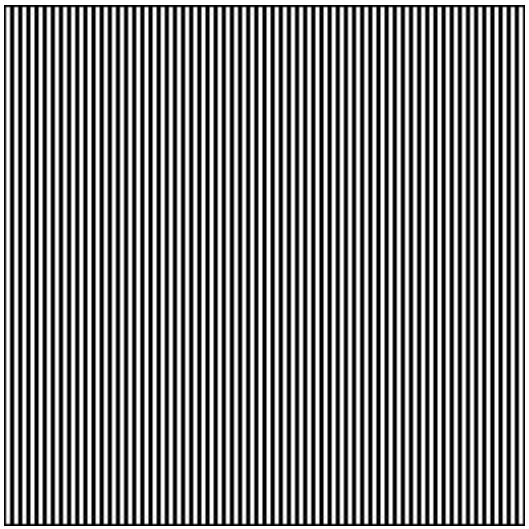
Filtering techniques typically modify frequency characteristics:

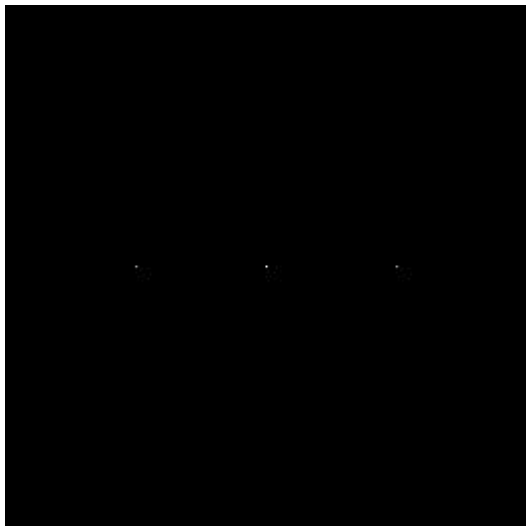
- Remove noise (decrease high frequencies)
- Smooth (decrease high frequencies, increase low frequencies)
- Enhance edges (increase medium frequencies)



The maximum frequency

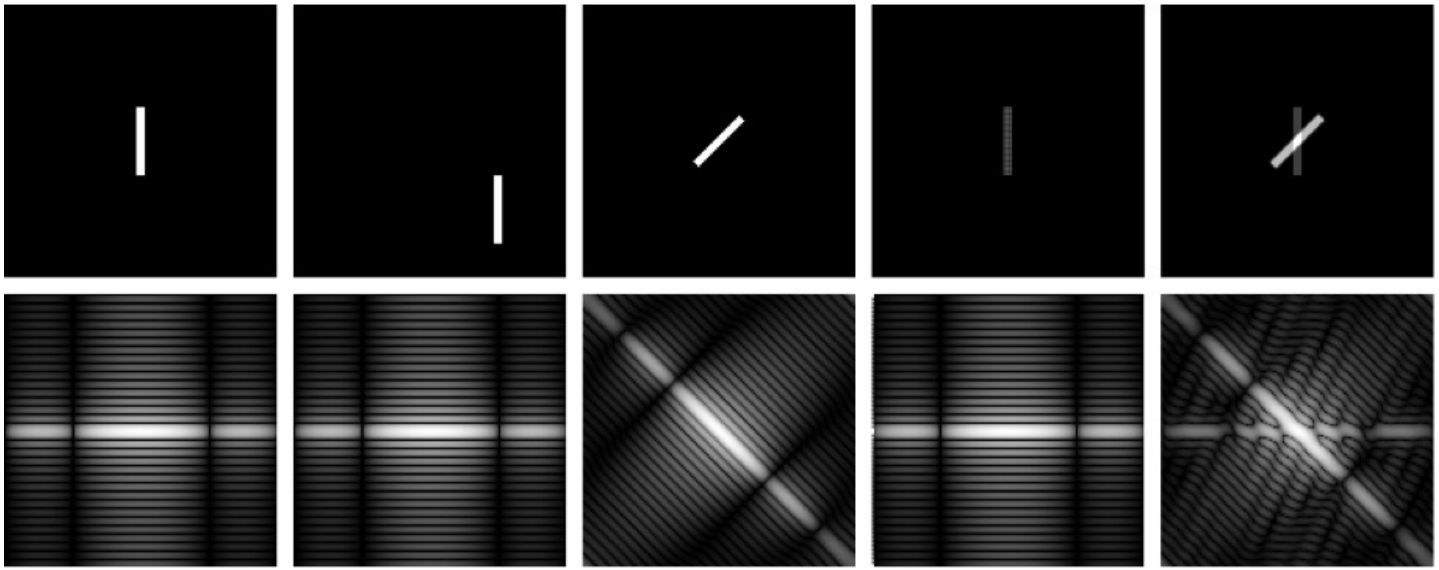
- The maximum frequency which can be represented in the spatial domain are one pixel wide stripes (period=2): $\omega_{max} = 2\pi\frac{1}{2} = \pi$
- So, 2 pixel wide stripes (period=4) give $\omega = 2\pi\frac{1}{4} = \frac{1}{2}\omega_{max}$





- Two points halfway between centre and the edge of the image, i.e. the represented frequency is half of the maximum.
- One point in the middle shows the DC-value (image mean).

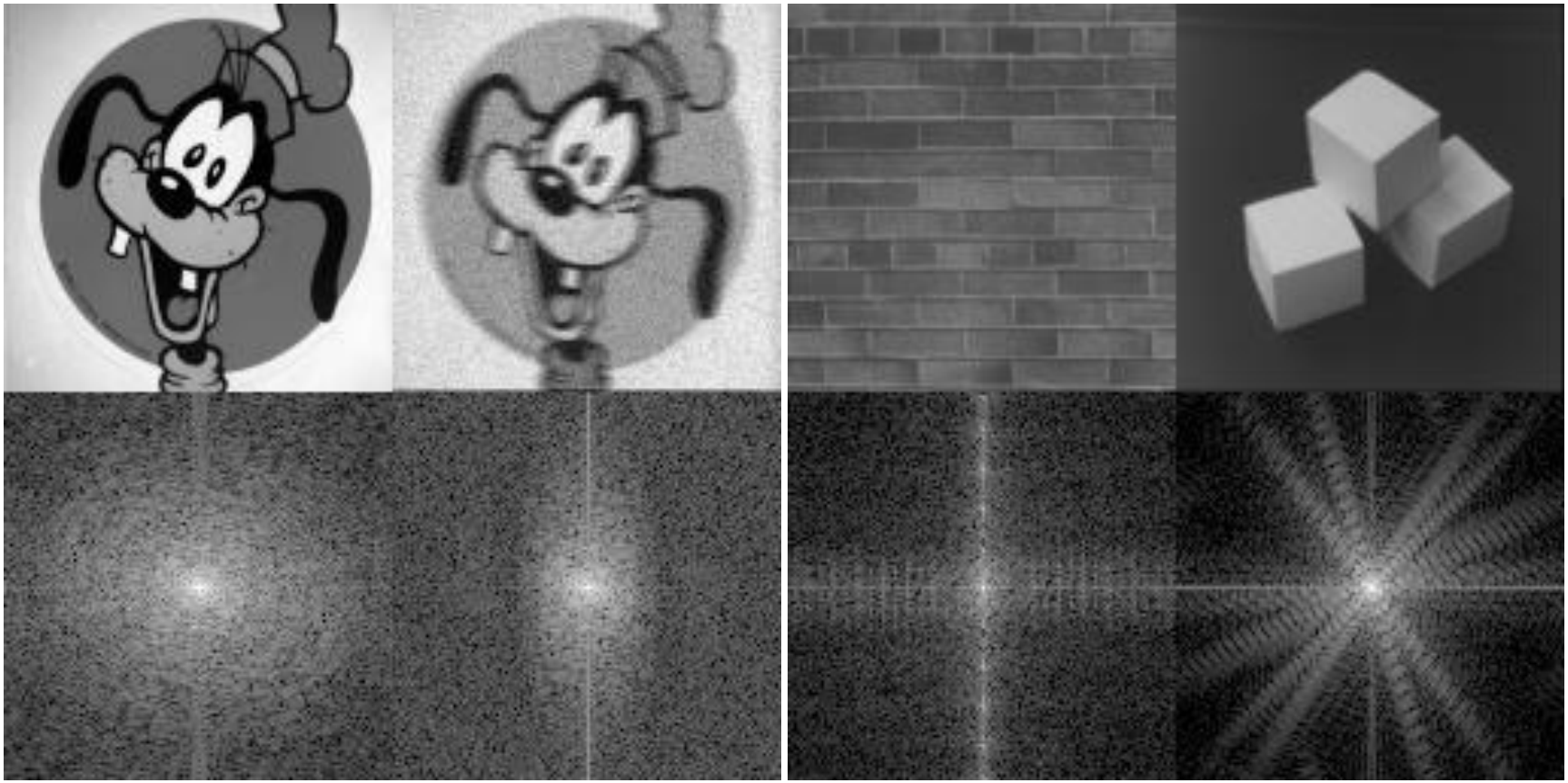
Example: Transformations



Observations:

- Translation only affects phase (not shown), not magnitude (shown)
- A rotation in one domain becomes a rotation in the other domain
- Fourier transform is a linear operation, $\mathcal{F}\big(af(x) + bg(x)\big) = \mathrm{a}\mathcal{F}\big(f(x)) + b\mathcal{F}(g(x)\big)$

Example images and Fourier transforms

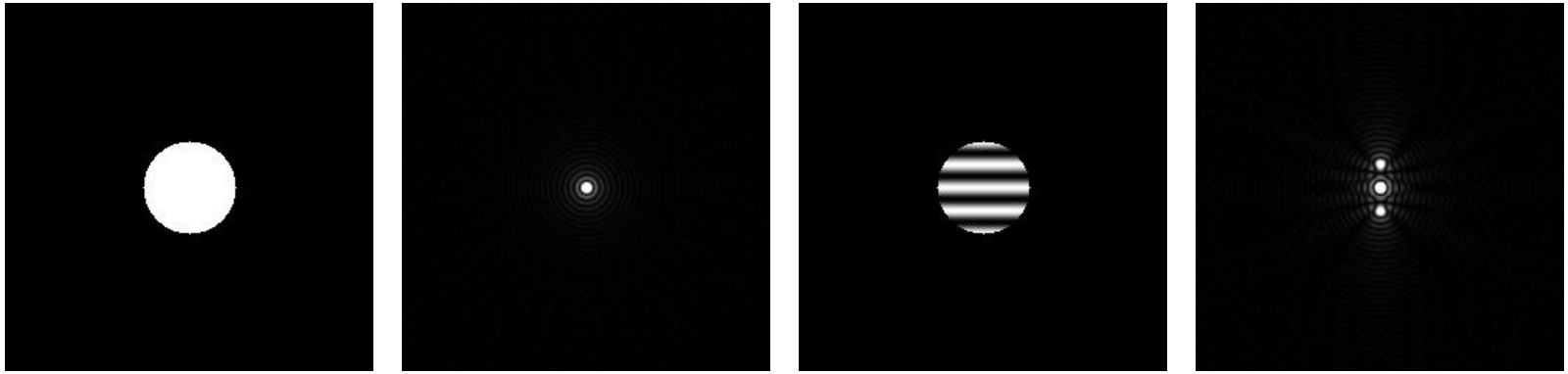


Property: Modulation

- If the original function is multiplied with an exponential like the one below and transformed, it will result in a shift of the origin of the frequency plane to (u_0, v_0).

$$\mathcal{F}\left[f(x,y)e^{2\pi i\left(\frac{xu_0}{M}+\frac{yv_0}{N}\right)}\right] = \hat{f}(u-u_0, v-v_0)$$

From left: Original image, magnitude of the Fourier spectrum, original multiplied by $1+\cos(\omega y)$ at a relative frequency of 16, magnitude of the Fourier spectrum.

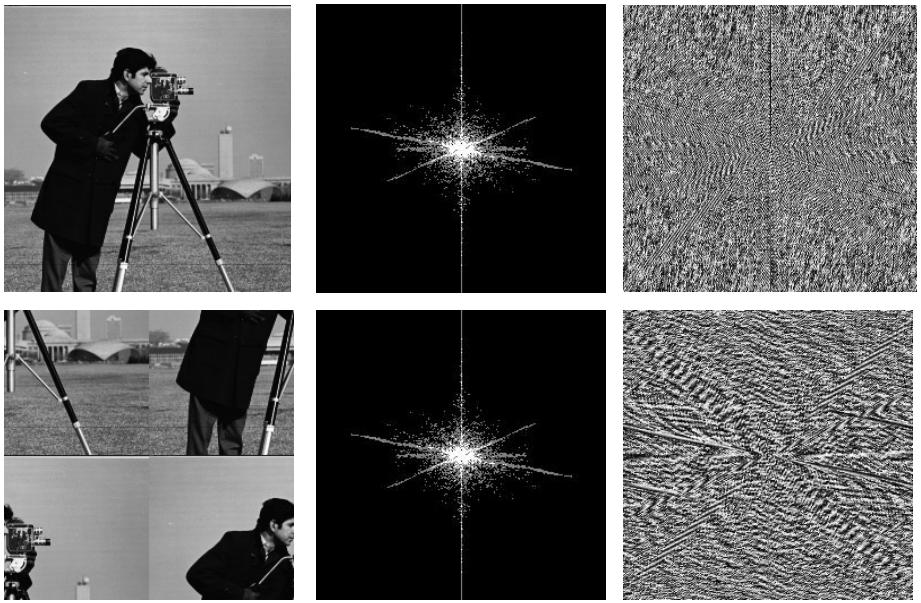


$$\text{Note}: 1+\cos\omega y = 1+\tfrac{1}{2}e^{i\omega y}+\tfrac{1}{2}e^{-i\omega y}.$$

Property: Translation

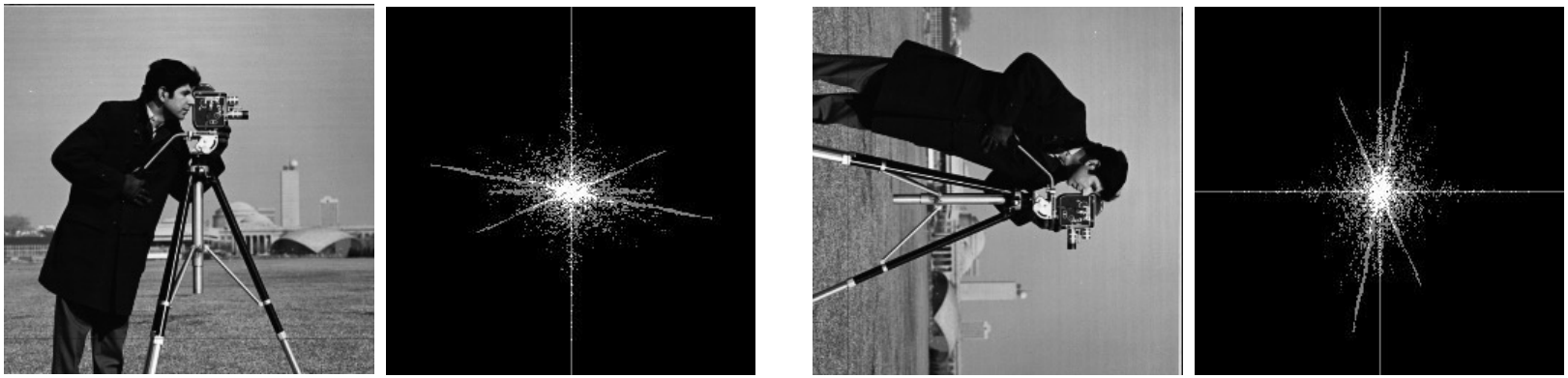
- (vice versa) If the image is moved, the Fourier spectrum undergoes a phase shift, but magnitude of the spectrum remains the same.

$$\mathcal{F}\left[\mathrm{f}(x-x_0, y-y_0)\right] = \hat{f}(u,v)e^{-2\pi i(\frac{x_0 u}{M}+\frac{y_0 v}{N})}, \quad |\hat{f}(u,v)e^{-2\pi i(\frac{x_0 u}{M}+\frac{y_0 v}{N})}| = |\hat{f}(u,v)|$$



Property: Rotation

- Rotation of the original image f rotates $\hat{f}$ by the same angle.



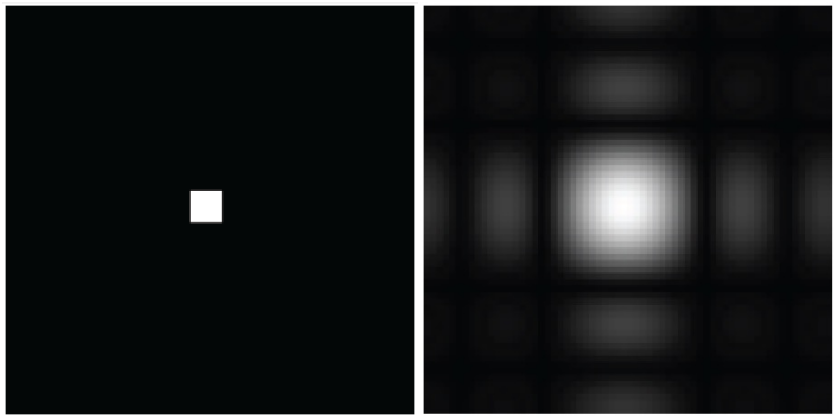
  - Exercise: Introduce polar coordinates and perform direct substitution.
  - Note: The rotation angle and phase angle are two very different kinds of angles.

Property: Scaling

- Compression (scale down) in spatial domain is same as expansion (scale up) in Fourier domain (and vice versa).

$$\mathrm{g}(x, y) = \mathrm{f}\left(S_x x, S_y y\right)$$

$$\hat{\mathrm{g}}(u, v) = \frac{1}{|S_x S_y|} \hat{\mathrm{f}}\left(\frac{u}{S_x}, \frac{v}{S_y}\right)$$



Property: Periodicity

- The DFT and its inverse are periodic with period N, for an $N \times N$ image. This means:

$$\hat{f}(u,v) = \hat{f}(u+N,v) = \hat{f}(u,v+N) = \hat{f}(u+N,v+N)$$

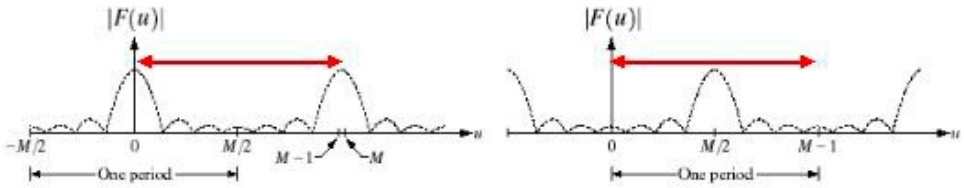
- This is easy to understand since

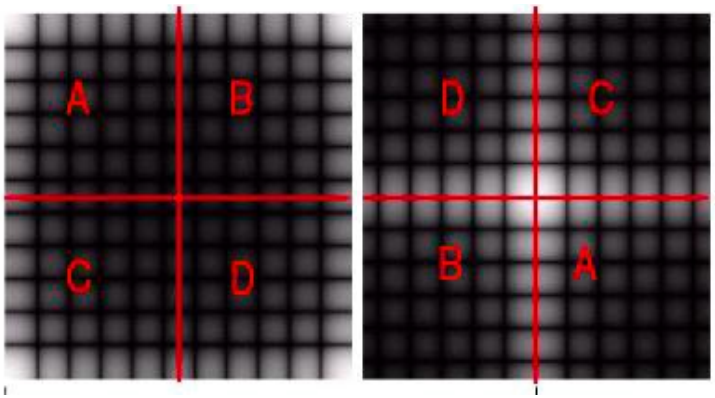
$$e^{-2\pi i(\frac{xu}{N})} = e^{-2\pi i(\frac{x(u+kN)}{N})}$$

- This property defines the implied symmetry in the Fourier spectrum as well as that $\hat{f}(u,v)$ repeats itself infinitely.
- However, only one period is enough to reconstruct the original image $f(x,y)$.

Property: Conjugate Symmetry

- The Fourier transform satisfies $\hat{f}(u,v) = \hat{f}^*(-u,-v)$ and $\left|\hat{f}(u,v)\right| = \left|\hat{f}^*(-u,-v)\right|$.
- With periodicity and above, we have that $\hat{f}(u,v)$ has period N and is (conjugate) symmetric around the origin.





- Thus we don't need $2N^2$ (N^2 real and N^2 imaginary) values to represent a $N \times N$ image in Fourier domain, but N^2 due to the symmetry.

Summary of good questions

- What properties does a linear shift-invariant filter have?
- How do you define a convolution?
- Why are convolutions important in linear filtering?
- How do you define a 2D Fourier transform?
- If you apply a Fourier transform to an image, what do you get?
- What information does the phase contain? And the magnitude?
- What is the Fourier transform of a convolution? Why important?
- What does separability of filters mean?
- How do you interpret a point in the Fourier domain in the spatial domain?
- How do you apply a discrete Fourier transform?
- What happens to the Fourier transform, if you translate or rotate an image?
- In what sense is the Fourier transform symmetric?

Recommended reading

- Gonzalez and Woods: Chapters 3.4, 4.3-4.7
- Szeliski: Chapter 3.2, 3.4
- Introduction to Lab 1