

Signal Processing for Images

Fourier and Convolution



Antonio Carta

2026-09-17

University of Pisa

antonio.cart@unipi.it

- Think of images as **signals**, and apply 1D/2D signal processing tools to them
- Build the tools (convolution, Fourier transform) that underlies filtering, sampling, compression, and CNN design

- images as signals
- convolution
- Fourier transform

Images as Signals

- An image is a function $f(x, y)$ sampled on a grid: $f[m, n]$, $m = 0, \dots, M - 1$, $n = 0, \dots, N - 1$
- Pixel values = samples of an underlying scene:
 - scene $\rightarrow$ optics $\rightarrow$ continuous image $\rightarrow$ sampled/quantized pixel grid
- Two complementary views:
 - **spatial domain**: translation-invariant transformations
 - **frequency domain**: images as superpositions of waves
- The two views are connected by the **convolution theorem**:
 - convolution in the spatial domain = pointwise multiplication in the frequency domain

Images as Superposition of Waves

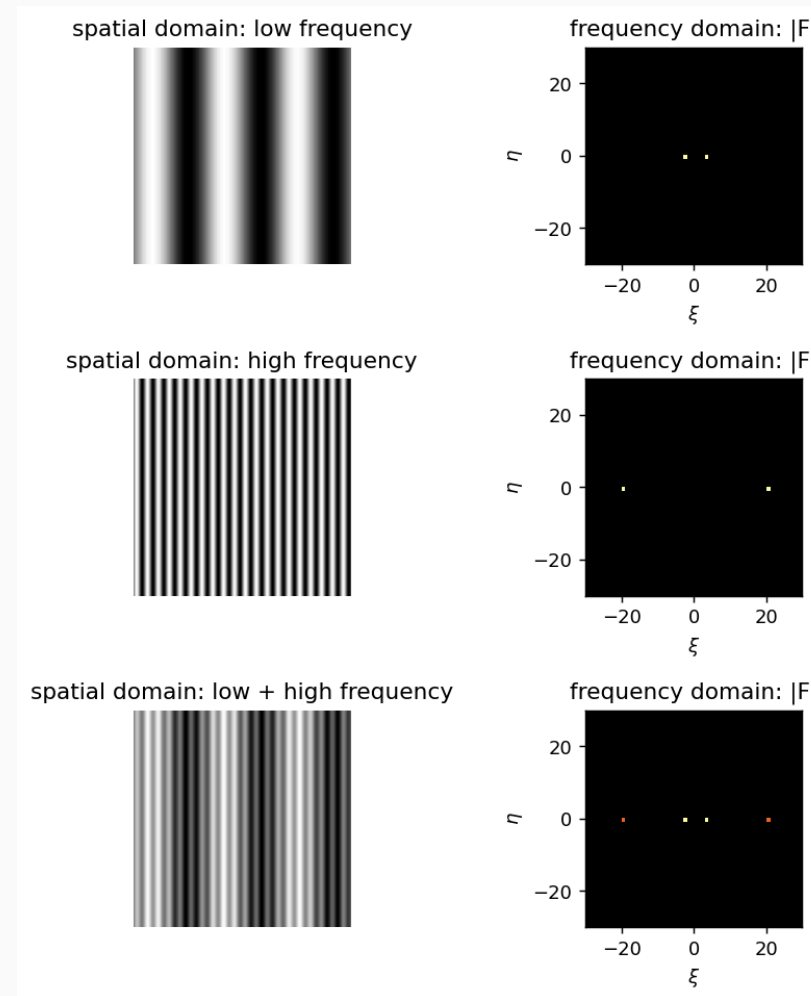


Figure 1: Spatial vs. frequency domain: pure low-frequency, pure high-frequency, and a mixture, alongside their Fourier magnitude spectra

- Same image, two equivalent representations connected by the Fourier transform:
 - **spatial domain** (pixel grid)
 - **frequency domain** (oscillations)
- CNNs operate in the spatial domain, but each convolutional filter has a **frequency response**, so the network implicitly performs frequency-selective processing
 - example: edges are high-frequency content
 - example: noise is (often) high-frequency content
- Switching from space to frequency can make certain operations (filtering) or properties (periodicity, scale) much easier to reason about
 - example: blur removes/attenuates high-frequency content
 - example: aliasing is high-frequency noise that is a consequence of some image operations (e.g. naive downsampling)

- While the frequency and spatial domain look very different, switching between the two representations (**Fourier Transform**) is a linear operation
 - it can be viewed as a **change of basis**: represent the signal in coordinates where the operation becomes simple
 - this basis can be very convenient: convolution becomes pointwise multiplication in the frequency space

- A bird is always a bird, no matter where it is in the image
- **Translation invariance** is a fundamental property of many vision tasks



Figure 2: source: visionbook and Fredo Durand.

- Many interesting operations are also **linear**: the output is a linear combination of inputs
- **which operations in the image below are linear?**
- Note: these images are in color, but most of what we see today applies to grayscale. We can treat each channel separately

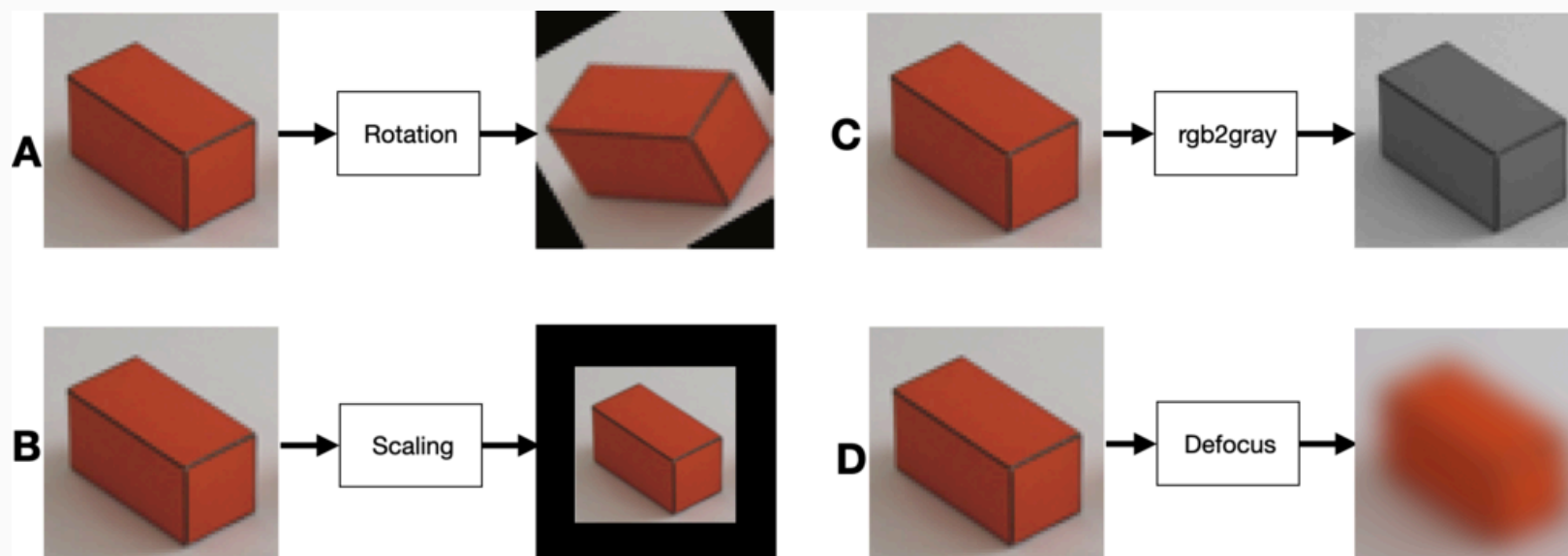


Figure 3: source: visionbook. Which operations can be implemented as convolutions?

Linear, Shift-Invariant (LSI) Systems

- A system T is **linear**: $T[af_1 + bf_2] = aT[f_1] + bT[f_2]$
- A system is **shift-invariant**: shifting the input shifts the output identically

$$l_{\text{out}}[n - n_0, m - m_0] = f(l_{\text{in}}[n - n_0, m - m_0])$$

- a convolution is an LSI system



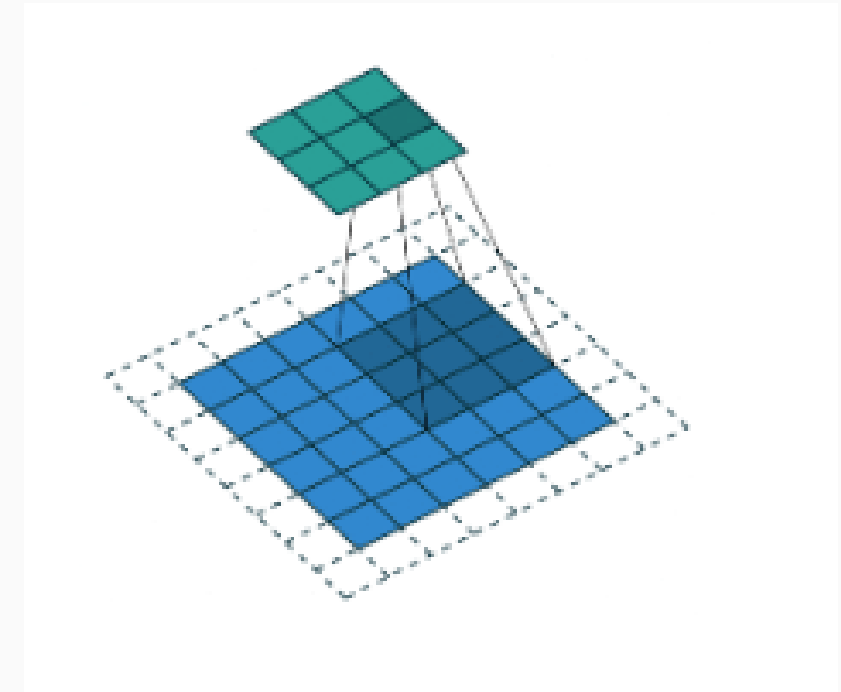
Figure 4: Examples of LSI image operators: blur, sharpen, edge filter

- images are signals
 - consequences (TBD): issues with discrete sampling
- LSI system are a simple class of operations that we can deeply analyze
 - (TBD) many common image preprocessing operations are LSI
 - we can analyze them in the spatial (convolution kernel) and frequency (Fourier) domains

Convolution

$$(f * h)[m, n] = \sum_i \sum_j f[i, j] h[m - i, n - j]$$

- we use the $*$ operator for convolution, not multiplication
- For finite kernels, sum is over the kernel's support (e.g. 3×3 , 5×5)
- This is the mathematical operation underlying image filtering **and** CNN “convolution” layers
 - in DNN we actually use cross-correlation (no kernel flip), but the two are equivalent for learned kernels



Source: vdumoulin/conv_arithmetic

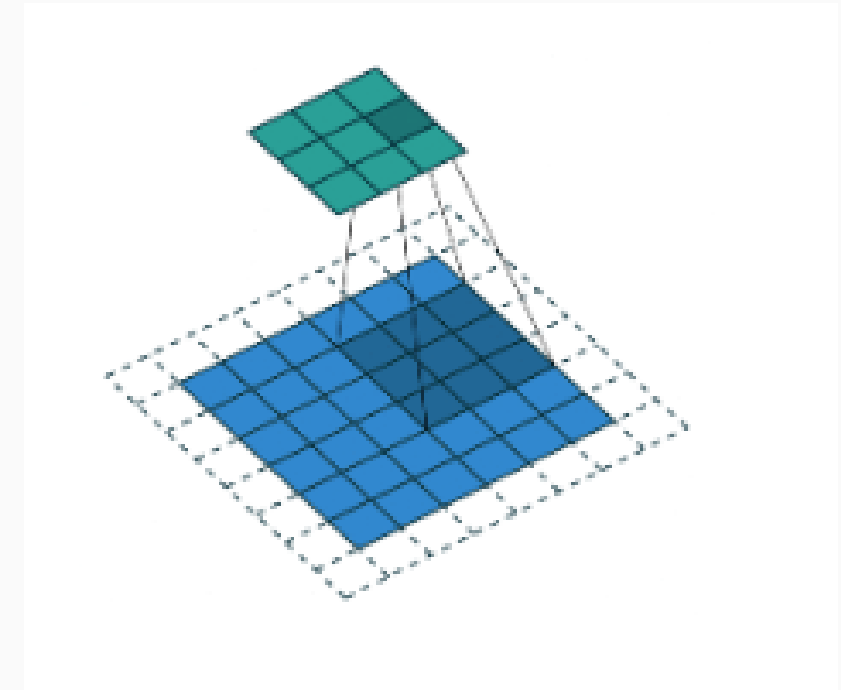
1D continuous convolution:

$$(f * h)(x) = \int_{-\infty}^{\infty} f(\tau)h(x - \tau) d\tau$$

- h is the **kernel**
- Output at x = weighted sum of nearby input values

2D continuous convolution:

$$(f * h)(x, y) = \int \int f(\tau, \sigma)h(x - \tau, y - \sigma) d\tau d\sigma$$



Source: vdumoulin/conv_arithmetic

Signal $f = [1, 2, 3, 4]$, kernel $h = [1, 0, -1]$, full convolution:

- Flip $h \rightarrow [-1, 0, 1]$
- Slide and compute dot products at each shift
- Result (full mode, zero-padded, length $4 + 3 - 1 = 6$):

$$f * h = [1, 2, 2, 2, -3, -4]$$

Interpretation: This kernel is an approximation of the discrete derivative. We have a large response where signal changes (edges), near zero in flat regions.

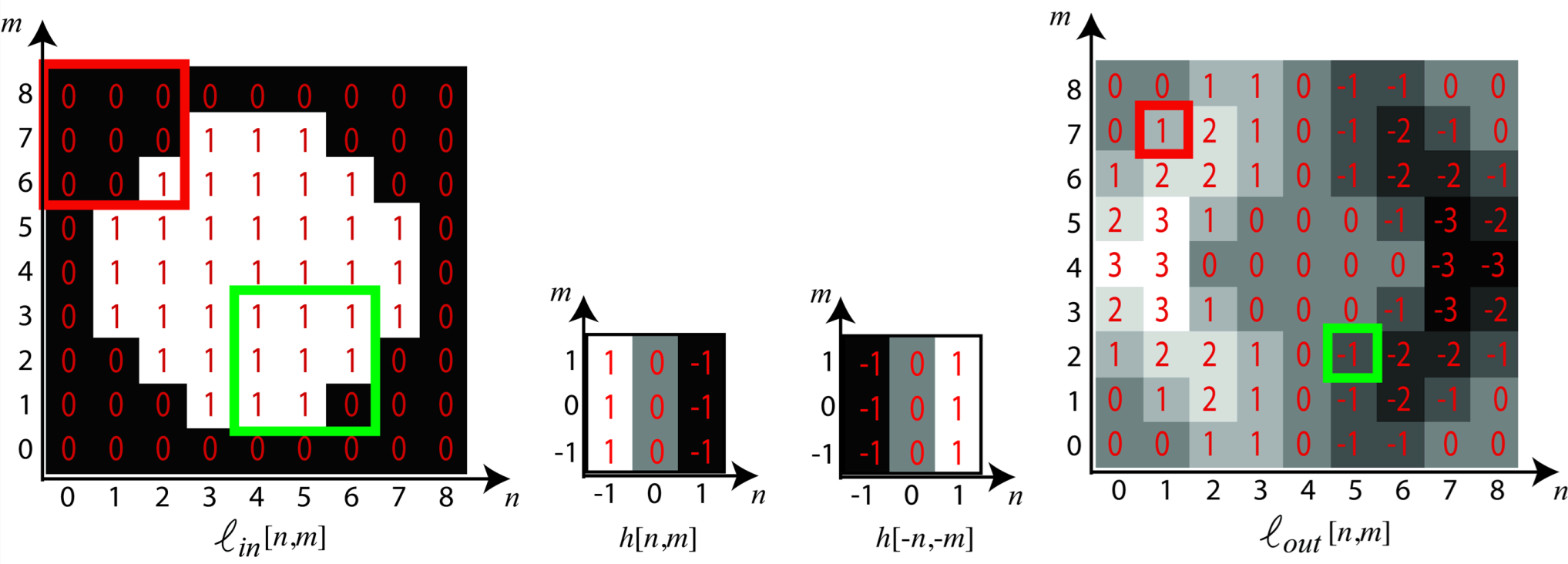


Figure 5: source: visionbook

What Operations Are LSI?

- Many image filters are LSI operations: blurring, sharpening, edge detection, any fixed convolution kernel
- Non-examples: thresholding, gamma correction (nonlinear), cropping/resizing (not shift-invariant), most CNN nonlinearities (ReLU, etc.)
- A system is LSI **iff** it can be written as $y = h * x$ for some fixed kernel h

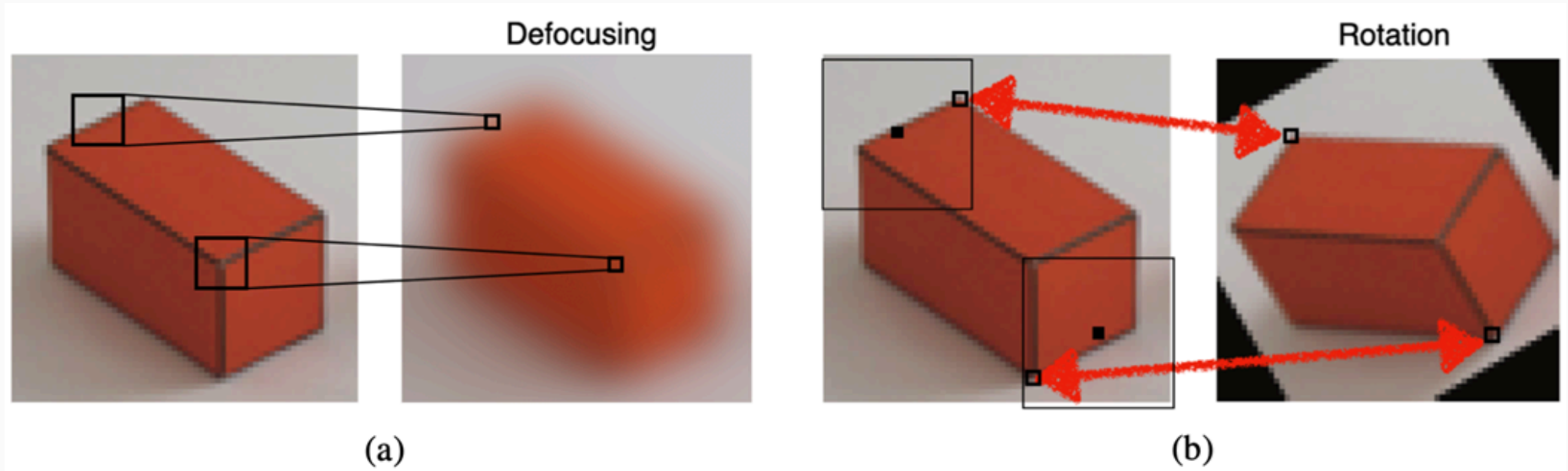


Figure 6: source: visionbook

- Blurring an image = replacing each pixel with a weighted average of its neighbors
- The same averaging rule is applied at every location (**shift-invariant**), and the result is a weighted sum of inputs (**linear**) $\Rightarrow$ blur is LSI, hence a convolution
- The “weights” pattern (e.g. a Gaussian) is exactly the convolution kernel h

2D Example: Gaussian Blur

- Gaussian kernel: $h(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$
- Convolution with a 5×5 Gaussian kernel ($\sigma = 2$)
- The result is a blurred image: sharp edges become a smooth ramp over a few pixels
- Larger σ results in wider kernel support and more smoothing

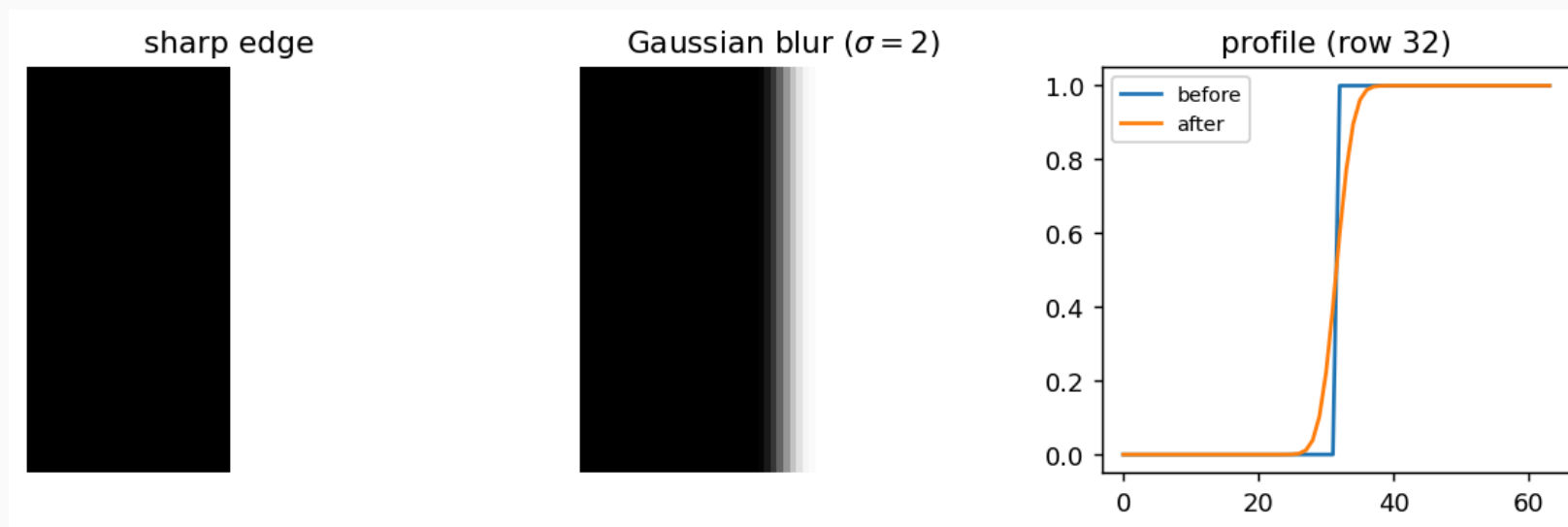


Figure 7: Sharp edge before and after Gaussian blur

Blur filters:

- **Box filter:** uniform averaging, simple blur

$$h = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

- **Gaussian:** $h(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$,
smooth, no ringing

Derivative approximations:

- **Sobel:**

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



Figure 8: Original image filtered with a box filter, a Gaussian, and a Sobel (gradient) filter

- **Linearity:** $f * (ah_1 + bh_2) = a(f * h_1) + b(f * h_2)$
- **Shift equivariance:** if we translate the input by (n_0, m_0) , the output translates by the same amount $l_{\text{out}}[n - n_0, m - m_0] = f(l_{\text{in}}[n - n_0, m - m_0])$
- **Commutativity:** $f * h = h * f$
- **Associativity:** $(f * h_1) * h_2 = f * (h_1 * h_2)$. We can chain filters by convolving kernels first
- **Identity:** the impulse $\delta[n]$ is the identity function ($\delta[n] * l[n] = l[n]$)

$$\delta[n] = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } n \neq 0 \end{cases}$$

- **Associativity:** $(f * h_1) * h_2 = f * (h_1 * h_2)$. We can convolve the **kernels** first, then apply once to the **image**
- Example: applying two 5×5 blurs sequentially = applying one equivalent 9×9 kernel (computed once)
- Used to fuse chains of filters into a single equivalent kernel

- A 2D kernel $h(x, y)$ is **separable** if $h(x, y) = h_x(x)h_y(y)$
 - h_x is a row vector, h_y is a column vector
- Gaussian is separable: $G_\sigma(x, y) = G_\sigma(x)G_\sigma(y)$
- Convolve rows then columns (1D passes) instead of full 2D convolution
- Cost: Given $M \times N$ image and $k \times k$ kernel, 2D convolution = $O(MNk^2)$, separable = $O(MNk + MNk) = O(MNk)$

Example:

$$\frac{1}{3} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \frac{1}{3} (1 \ 1 \ 1) = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Common implementation trick used in DNNs designed for low-resource scenarios like MobileNet and EfficientNet.

- **Standard conv**: each output channel mixes all input channels with a full $k \times k \times C_{\text{in}}$ kernel
- **Depthwise separable**:
 - (1) a $k \times k$ **depthwise** conv applied independently per channel
 - (2) a 1×1 **pointwise** conv to mix channels
- Reduces parameters/FLOPs: $O(k^2 C + C^2)$ vs. $O(k^2 C^2)$

Image edges have no neighbors outside the border. Common strategies:

- **Zero padding**: treat outside as 0 (can darken edges)
- **Replicate (clamp)**: repeat edge pixels
- **Reflect (mirror)**: mirror pixels across the border
- **Circular (wrap)**: treat image as periodic, natural for DFT-based convolution
- Choice affects visual artifacts

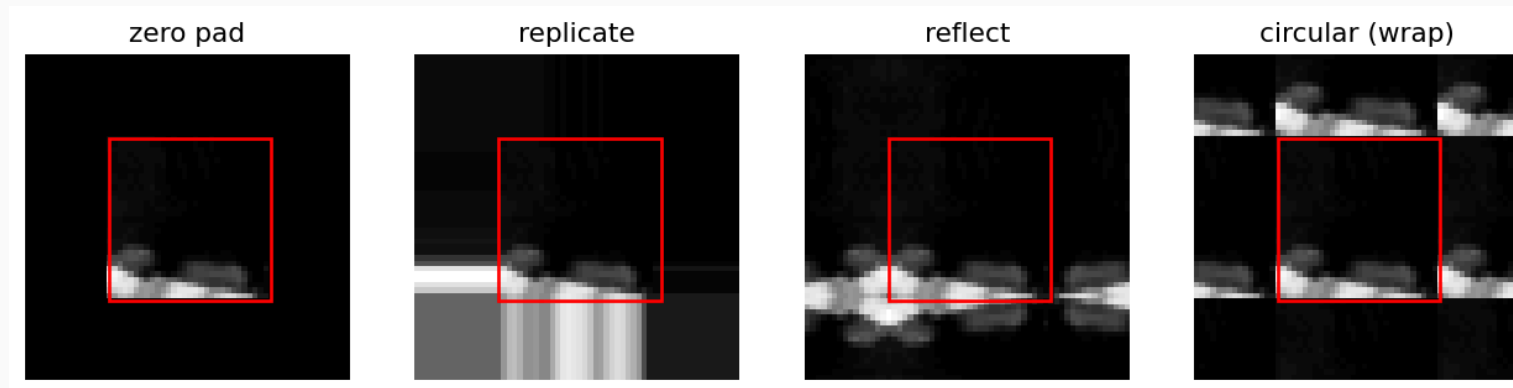


Figure 9: The same image patch padded with each strategy (red box = original region)

Cross-Correlation vs. Convolution

- **Convolution:** $(f * h)[m, n] = \sum_{i,j} f[i, j]h[m - i, n - j]$, kernel is flipped before sliding
- **Cross-correlation:** $(f \star h)[m, n] = \sum_{i,j} f[i, j]h[m + i, n + j]$, no kernel flip
- For symmetric kernels (Gaussian, box), the two are identical
- **CNNs implement cross-correlation** because the difference doesn't matter for learned kernels (the network can learn the flip), but it matters for hand-designed filters and theory

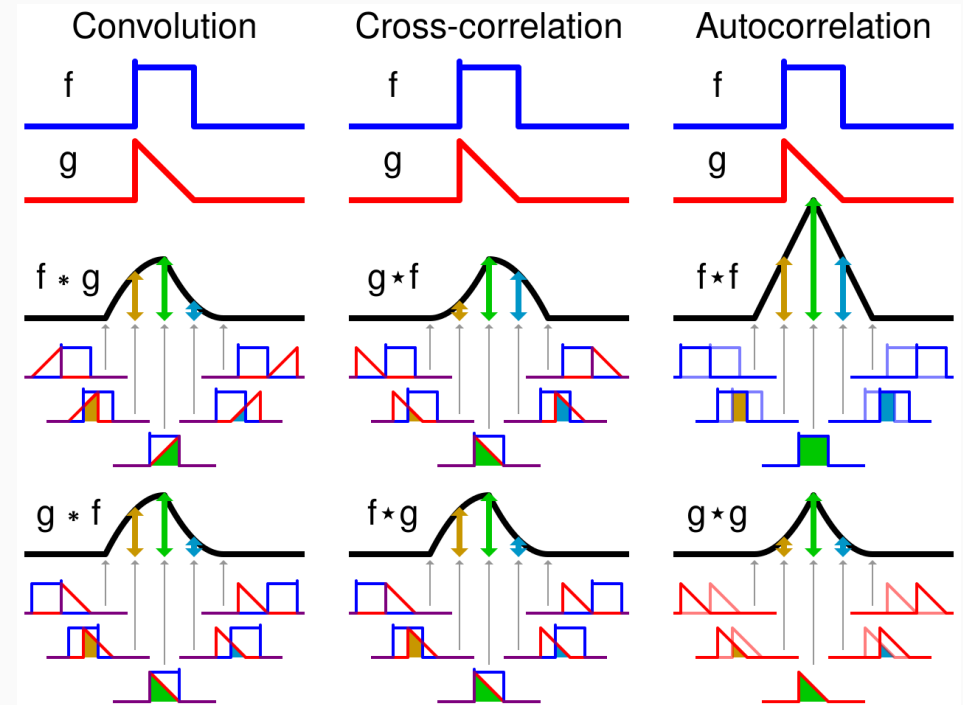


Figure 10: Convolution vs. cross-correlation of the same signal and kernel, note the flipped kernel in convolution. Convolution is commutative, cross-correlation is not. source: Wikimedia

- Discrete convolution with a fixed kernel h is a **linear operator** on the vector of pixel values (flattened image)
- For 1D circular convolution, this matrix is a **circulant (Toeplitz-like) matrix** H :

$$y = Hx, \quad H_{ij} = h[(i - j) \bmod N]$$

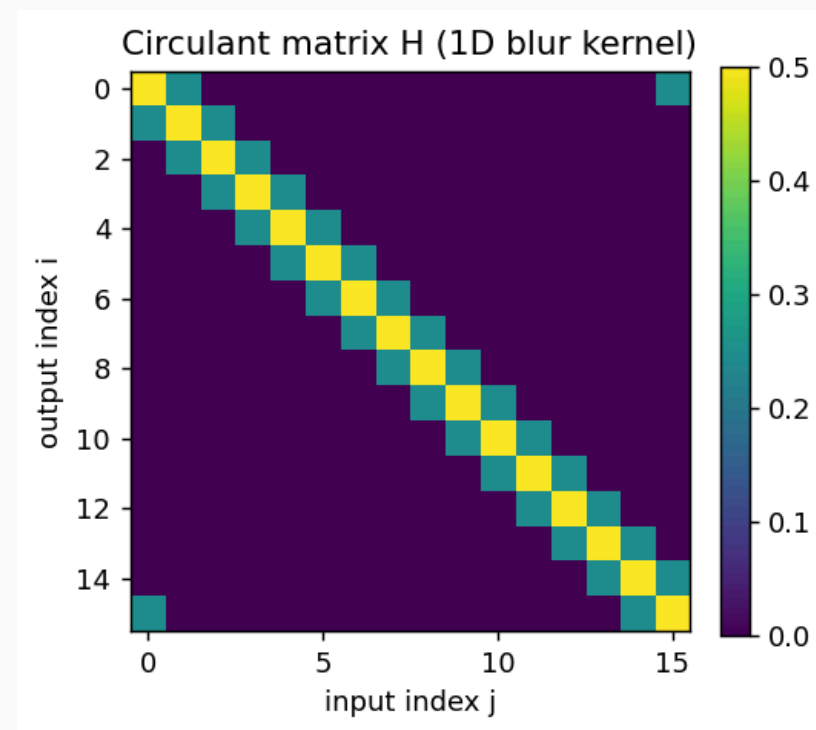


Figure 11: A circulant matrix implementing a 1D blur kernel

Example: Convolution Matrix

- Signal $x = [1, 2, 3, 4]^T$, circular kernel $h = [1, -1, 0, 0]$
- Circulant matrix H has rows that are cyclic shifts of h :

$$H = \begin{pmatrix} 1 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

- $Hx = [1 - 4, 2 - 1, 3 - 2, 4 - 3]^T = [-3, 1, 1, 1]^T$, matches circular convolution of x with h

- **Convolution** is the operation implemented by every LSI system, parameterized by a kernel h
- **Properties** (linearity, commutativity, associativity, separability) let us design and combine filters efficiently
- **Equivalent views**: sliding weighted sum, matrix multiplication by a circulant matrix.
 - Next slides: convolution as multiplication in the frequency domain
- **Equivariance** to translation is the first kind of invariance that we solved.
 - We are still missing invariance to scale, rotation, and other transformations. TBD in the next lectures.

Fourier Transform

- A **change of representation** can reveal structure and make certain operations easier
- Example: switching from euclidean coordinates to polar coordinates makes circular patterns easier to analyze
- The **Fourier transform** is a change of representation from the spatial domain to the frequency domain, where signals are represented as superpositions of sinusoids
- It is a linear change of basis
 - $x = Hl_{\text{in}}$
 - $l_{\text{in}} = H^{-1}x$

Continuous Sine Wave

$$s(t) = A \sin(\omega t - \theta)$$

- A amplitude
- ω angular frequency (radians/s)
- θ phase (radians)
- period $T = 2\frac{\pi}{\omega}$

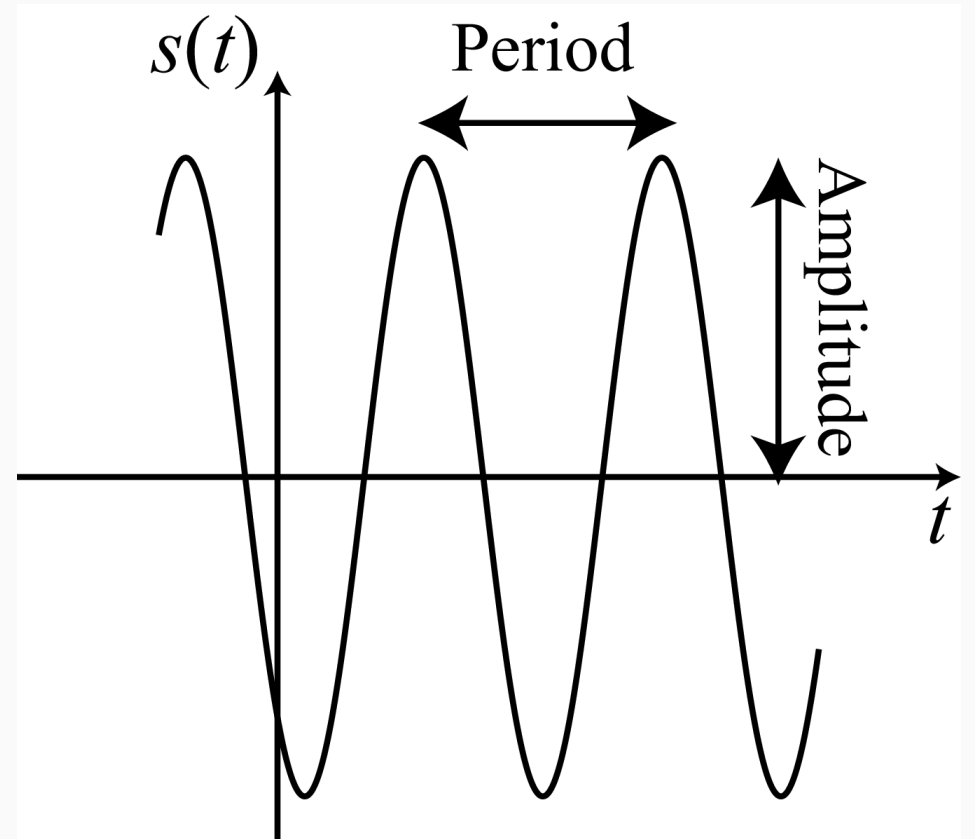


Figure 12: source: visionbook

Fourier series: Any 1D periodic signal $l(t)$ can be represented as a sum of sinusoids for $t \in [0, \pi]$

$$l(t) = a_1 \sin(t) + a_2 \sin(2t) + a_3 \sin(3t) + \dots$$

where a_k

$$a_k = \frac{1}{T} \int_0^T l(t) \sin(kt) dt$$

- Switching to the frequency domain makes it easier to analyze and manipulate signals, especially for filtering and compression
- A nice visual reference can be found at <https://www.3blue1brown.com/lessons/fourier-transforms/>

Fourier Series: Decomposing Periodic Signals

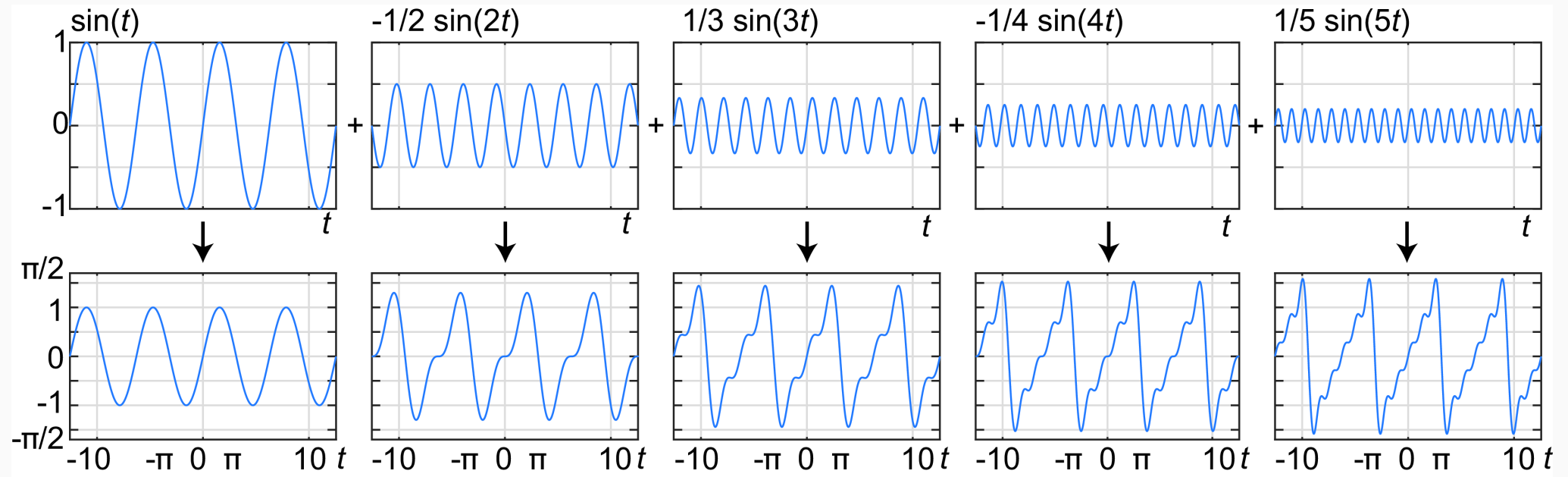


Figure 13: Sawtooth signal with its Fourier series approximation. source: visionbook

Thinking in terms of frequencies helps to understand vision operations from a different viewpoint.

In frequency terms:

- A **blur** is removing high frequencies
- **Edges** are high-frequency content
- **Noise** often spreads across frequencies differently than signal
- We can **compress** images using techniques like JPEG, which operate by discarding imperceptible high frequencies
- **Aliasing** is a frequency-domain phenomenon

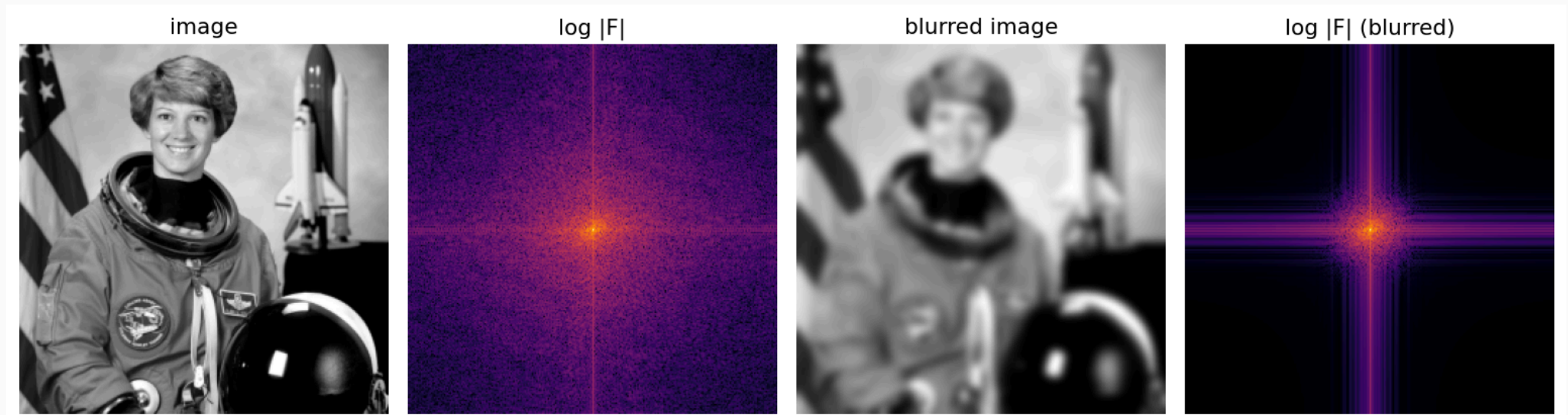


Figure 14: An image and its log-magnitude spectrum, before and after blurring: blur removes high-frequency (outer) content

Discrete Sine Wave

$$s[n] = A \sin(\omega n - \theta)$$

- n discrete time index
- the discrete signal $s[n]$ is not periodic in general
- it is periodic only if ω is a rational multiple of 2π , it will be periodic with some period N (e.g. $\omega = 2\pi \frac{k}{N}$)

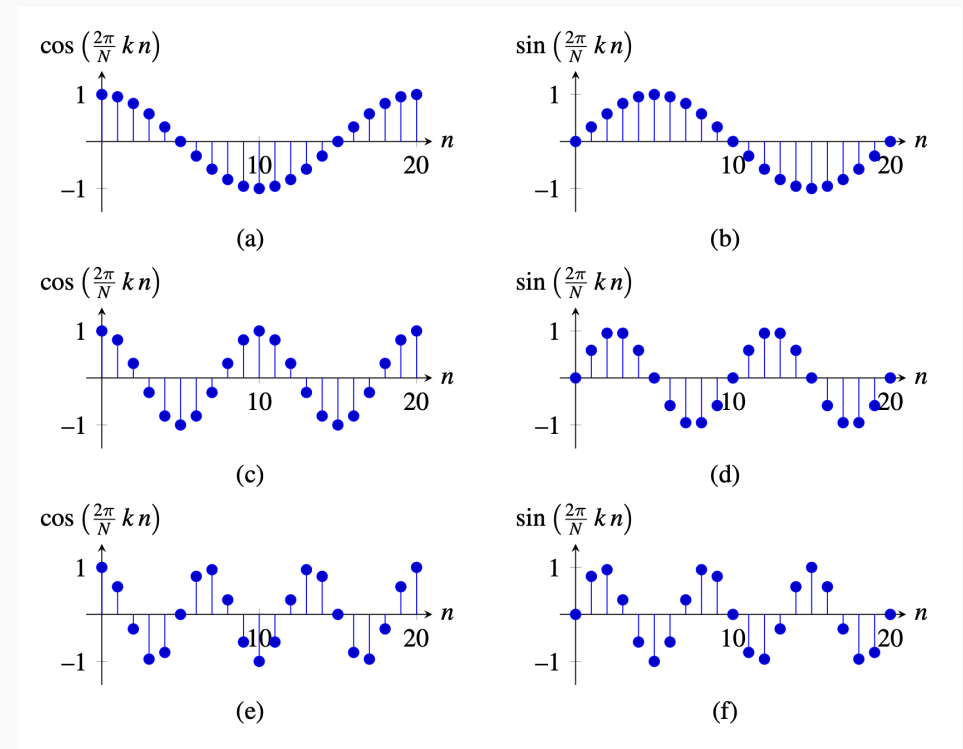


Figure 15: source: visionbook

Discrete Sine Wave

- to make the period explicit we use sine and cosine waves of the form

sine wave

$$s[n] = \sin\left(\frac{2\pi}{N}kn - \theta\right)$$

cosine wave

$$c[n] = \cos\left(\frac{2\pi}{N}kn - \theta\right)$$

- N is the length of the signal
- $k \in \left[1, \frac{N}{2}\right]$ is the frequency as the number of cycles in the signal

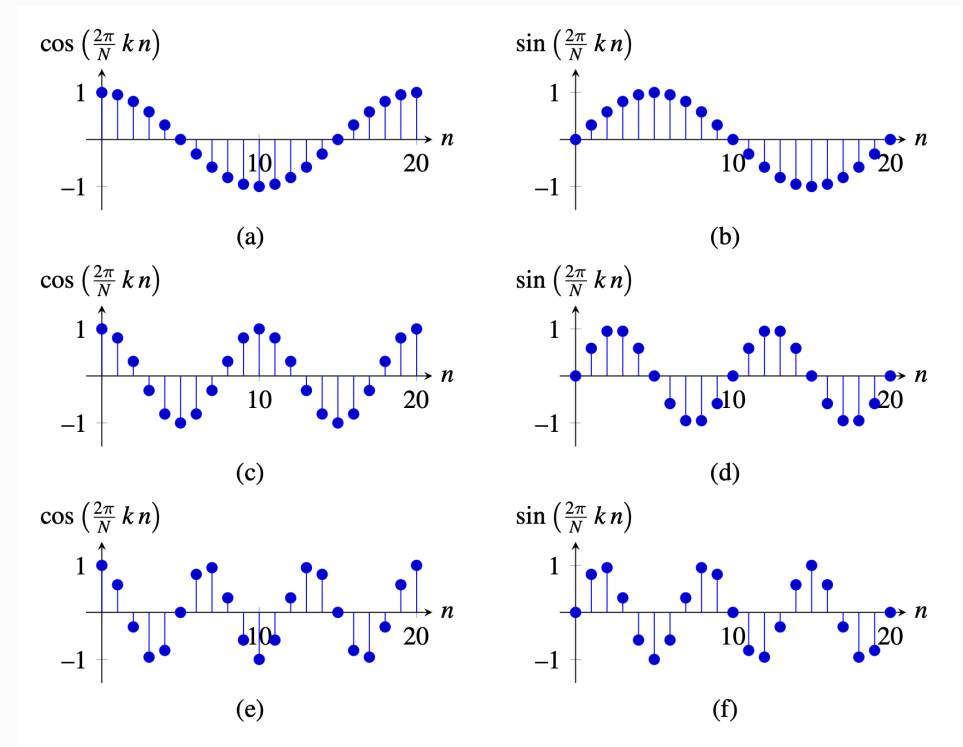


Figure 16: source: visionbook

- given N the length of the signal
- $c_{N-k} = c_k$
- $s_{N-k} = -s_k$
- $k = 0$ is the DC component (constant signal)
- $k = \frac{N}{2}$ is the Nyquist frequency (highest representable frequency, oscillates between positive and negative values at each sample)
 - we will clarify this statement later

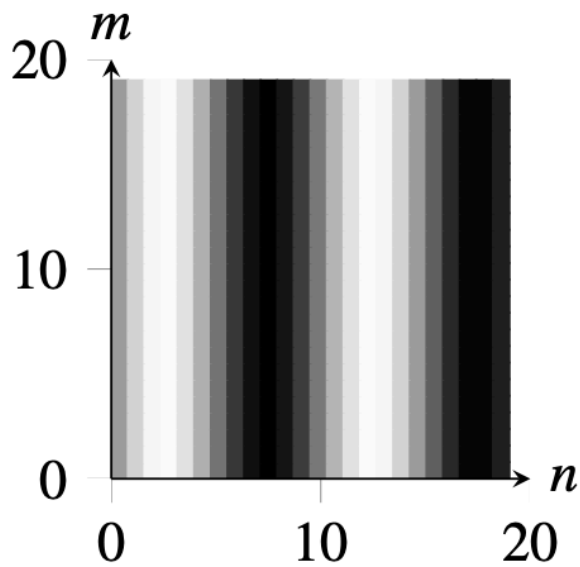
sine wave

$$s[n] = \sin\left(\frac{2\pi}{N}kn - \theta\right)$$

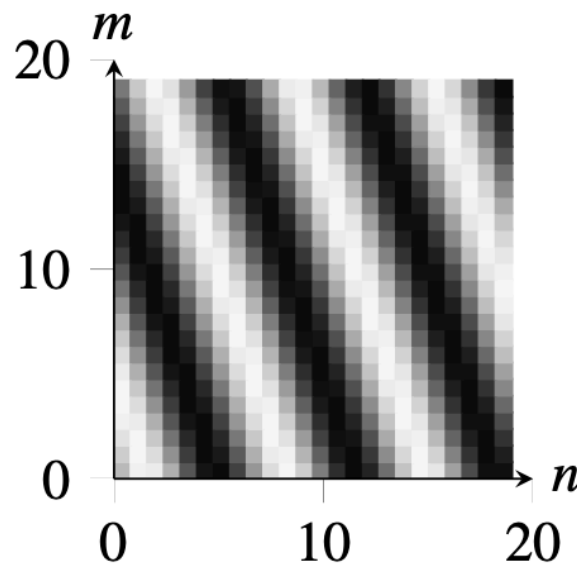
cosine wave

$$c[n] = \cos\left(\frac{2\pi}{N}kn - \theta\right)$$

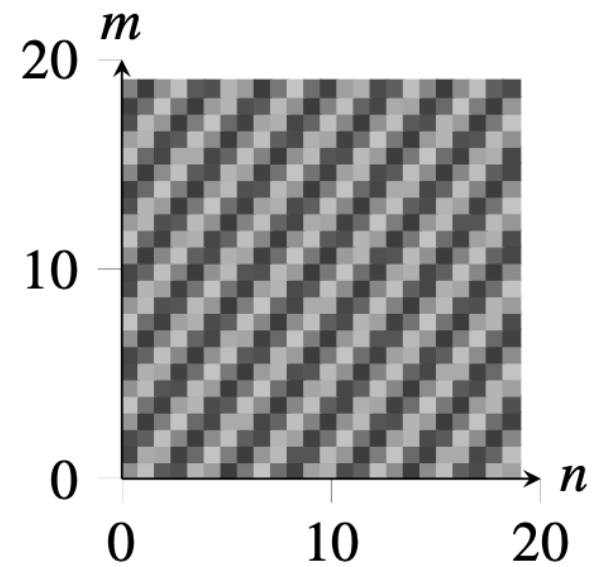
$$s_{u,v}[m, n] = A \sin\left(2\pi\left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$



(a)



(b)



(c)

Figure 17: source: visionbook

Rotations in the Complex Plane

- Euler's formula: $e^{i\theta} = \cos \theta + i \sin \theta$
- $|e^{i\theta}| = 1$: a point on the unit circle, at angle θ
- Multiplying by $e^{i\theta}$ = rotating by θ (no change in magnitude)

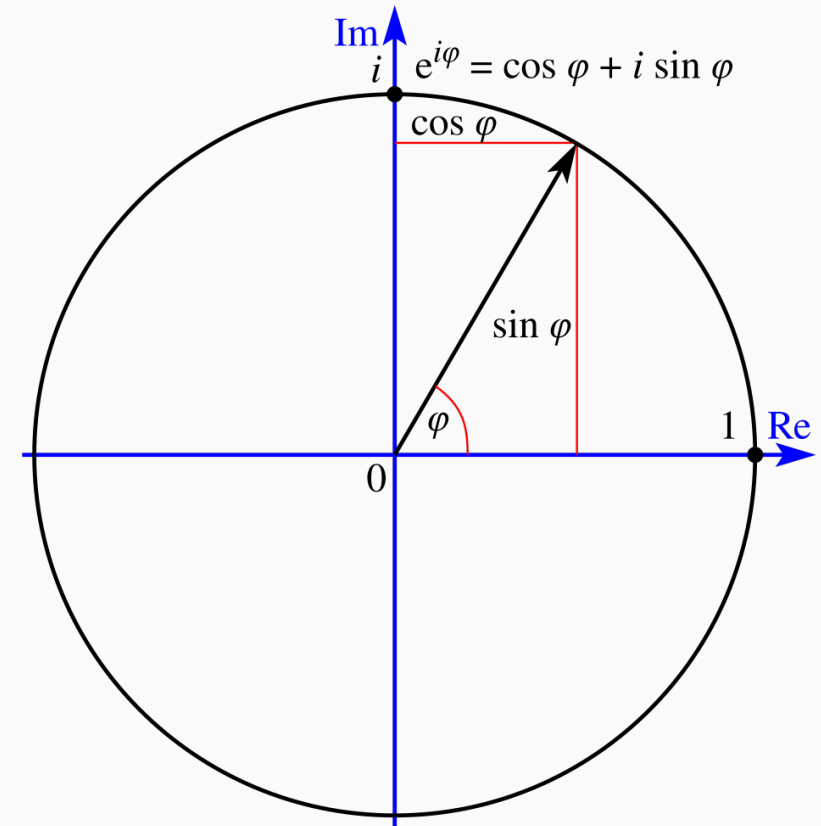


Figure 18: Complex exponentials as rotations in the complex plane.

source: wikimedia

Why do we bother with complex exponentials instead of just $\sin / \cos$? Mathematical convenience:

- Complex exponentials turn shifts, derivatives, and convolutions into simple algebraic operations (multiplication by a phase or a scalar)
- Any complex number decomposes as magnitude + phase, a convenient representation for oscillations

$$e_u[n] = \exp\left(2\pi j \frac{un}{N}\right)$$

- Note: visionbook uses j instead of i for the imaginary unit, which is common in engineering contexts.
- $e_u[n]$ is related to sine and cosine waves by Euler's formula: $\exp(ja) = \cos a + j \sin a$

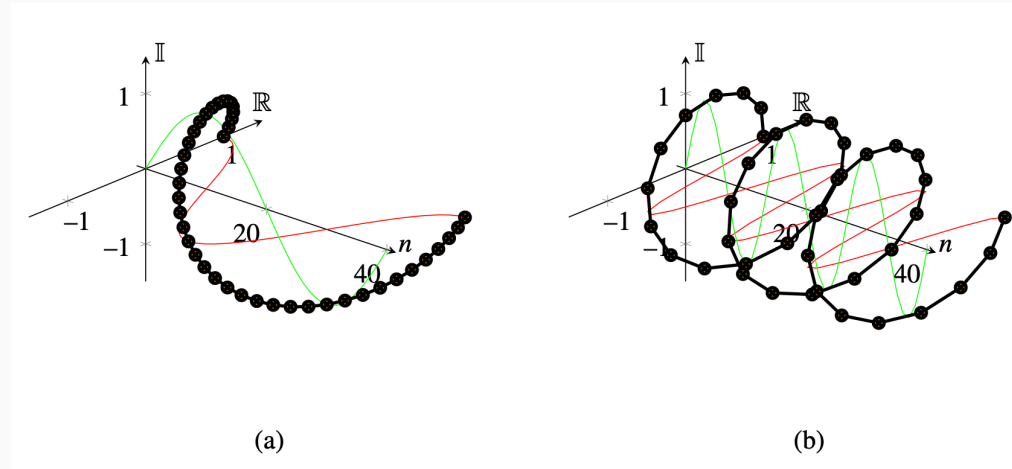


Figure 19: Complex exponentials as rotations in the complex plane. source: visionbook

$$e_{u,v}[m, n] = \exp\left(2\pi j\left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$

- u and v control the frequency in the horizontal and vertical directions, respectively
- the signal is separable: $e_{u,v}[m, n] = e_u[n]e_v[m]$

$$e_{u,v}[m, n] = \exp\left(2\pi j\left(\frac{un}{N} + \frac{vm}{M}\right)\right)$$

complex exponentials form an orthogonal basis because

$$\langle e_{u,v}, e_{u',v'} \rangle = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} e_{u,v}[n, m] e_{u',v'}^*[n, m] = MN \delta_{[u-u']} \delta_{[v-v']}$$

- where $\delta_x = \begin{cases} 1 & \text{if } x=0 \\ 0 & \text{o.w.} \end{cases}$ is the Kronecker's delta

Consequences

- any finite length discrete image can be decomposed as a linear combination of complex exponentials
- the Fourier basis is an orthogonal basis
 - the coefficients are unique
 - the coefficients are found via dot products

Example: Approximating a Square Wave

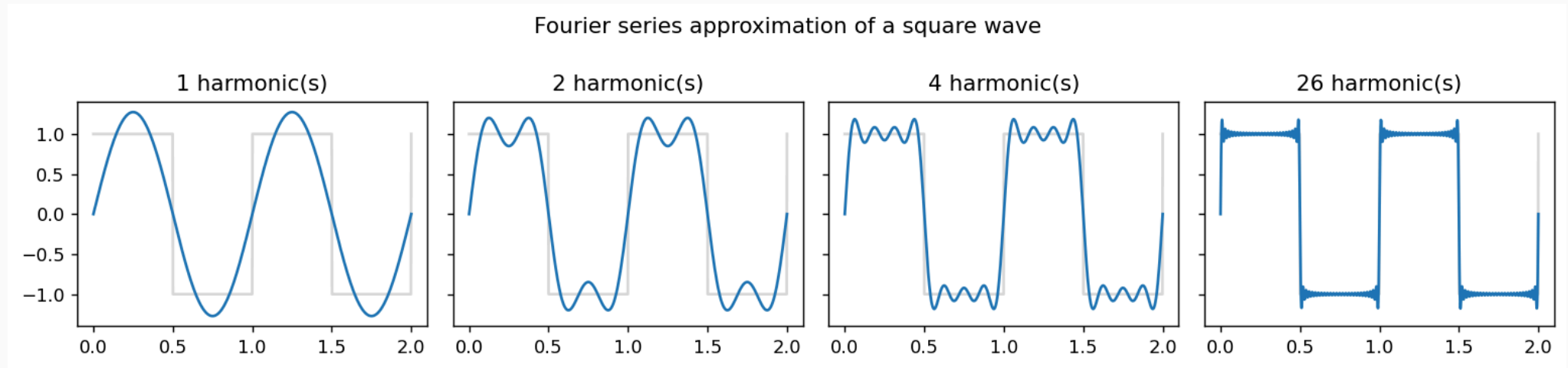


Figure 20: A square wave approximated by 1, 2, 4, and 26 odd harmonics

- More harmonics = better approximation, but a sharp jump always produces overshoot near the discontinuity (**Gibbs phenomenon**)
- High frequencies are responsible for sharp transitions

Given an image $l[n, m]$ of size $N \times M$, its **Discrete Fourier Transform (DFT)** is

$$\mathcal{L}[u, v] = \mathcal{F}\{\ell[n, m]\} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \ell[n, m] \exp\left(-2\pi j \left(u \frac{n}{N} + v \frac{m}{M}\right)\right)$$

The **Inverse DFT** is

$$\ell[n, m] = \mathcal{F}^{-1}\{\mathcal{L}[u, v]\} = \frac{1}{NM} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} \mathcal{L}[u, v] \exp\left(2\pi j \left(u \frac{n}{N} + v \frac{m}{M}\right)\right)$$

- $\mathcal{F}\{\ell[n, m]\}$ represents the image as a sum of complex exponentials at different spatial frequencies weighted by its coefficient
- Notice the special **DC frequency** (0, 0) is a constant value

- $\mathcal{L}[u, v]$ is periodic
 - $\forall a, b \in \mathbb{Z} : \mathcal{L}[u + aN, v + bM] = \mathcal{L}[u, v]$.
- given that $e_{N-u, M-v} = e_{-u, -v}$, we can write the Fourier transform with frequencies in $[-\frac{N}{2}, \frac{N}{2}]$ and $[-\frac{M}{2}, \frac{M}{2}]$
- this is how we show it in the plots (0 in the center). `fftshift` in numpy does this operation.

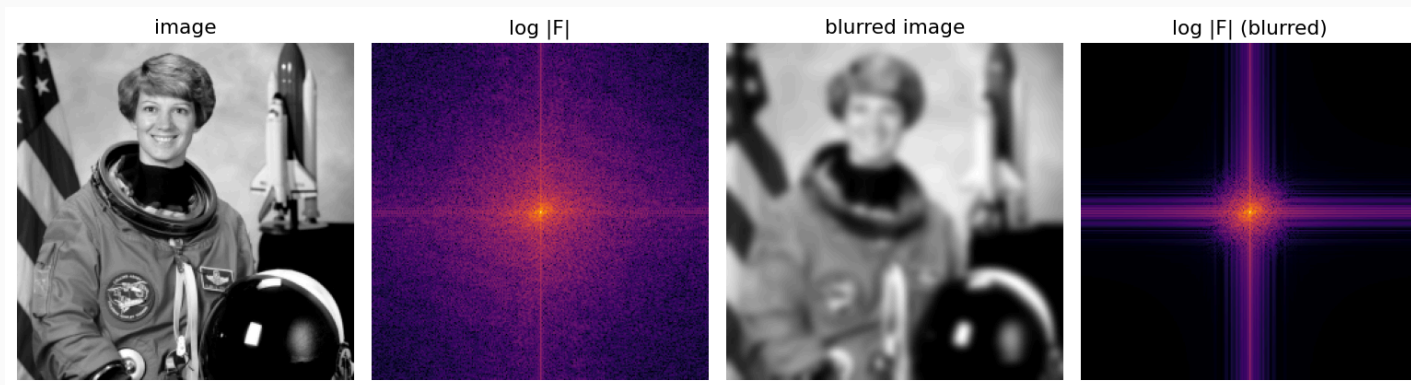


Figure 21: An image and its log-magnitude spectrum, before and after blurring: blur removes high-frequency (outer) content

$$\mathbf{F} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \exp\left(-2\pi j \frac{1}{N}\right) & \dots & \exp\left(-2\pi j \frac{N-1}{N}\right) \\ 1 & \exp\left(-2\pi j \frac{2}{N}\right) & \dots & \exp\left(-2\pi j \frac{2(N-1)}{N}\right) \\ \vdots & \ddots & \ddots & \vdots \\ 1 & \exp\left(-2\pi j \frac{N-1}{N}\right) & \dots & \exp\left(-2\pi j \frac{(N-1)^2}{N}\right) \end{pmatrix}$$

- the entries $F_{u,n} = \exp\left(-2\pi j \frac{un}{N}\right)$
- $\mathbf{F}$ is symmetric
- $\mathbf{F}^{-1} = \mathbf{F}^*$ (conjugate)
- the **Fast Fourier Transform** exploits symmetries in the structure of $\mathbf{F}$ to compute the DFT in $O(n \log n)$ instead of $O(n^2)$

Examples of Signals and their Fourier Transform

- the DFT returns complex numbers
- we can show image and real components separately

$$\mathcal{L}[u, v] = \text{Re}\{\mathcal{L}[u, v]\} + j \text{Im}\{\mathcal{L}[u, v]\}$$

- we usually decompose them into magnitude and phase
 - often we use the log-magnitude

$$\mathcal{L}[u, v] = A[u, v] \exp(j\theta[u, v])$$

Visualizing the Fourier Transform

FA 2018/19, Prof. Dr. J. J. D.

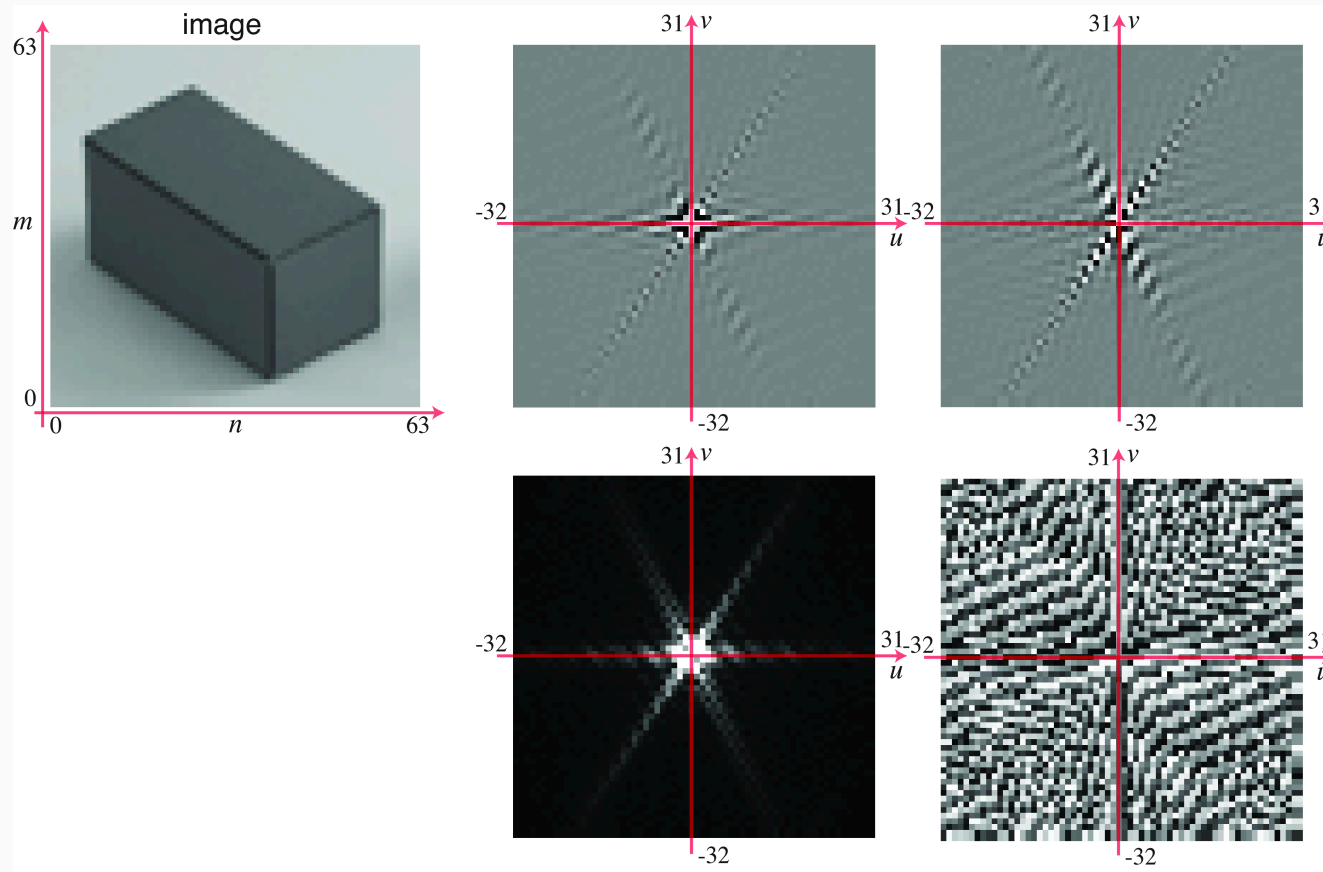


Figure 22: (top) the real and imaginary components, and (bottom) the amplitude and phase.
source: visionbook.

- **Smooth region**: low-frequency content
- **Sharp edge**: broadband with many frequencies, including very high
- **Periodic texture** (e.g. brick wall): energy concentrated at specific frequencies
- **conclusion**: a complex texture or feature can have a distinctive signature in the frequency domain

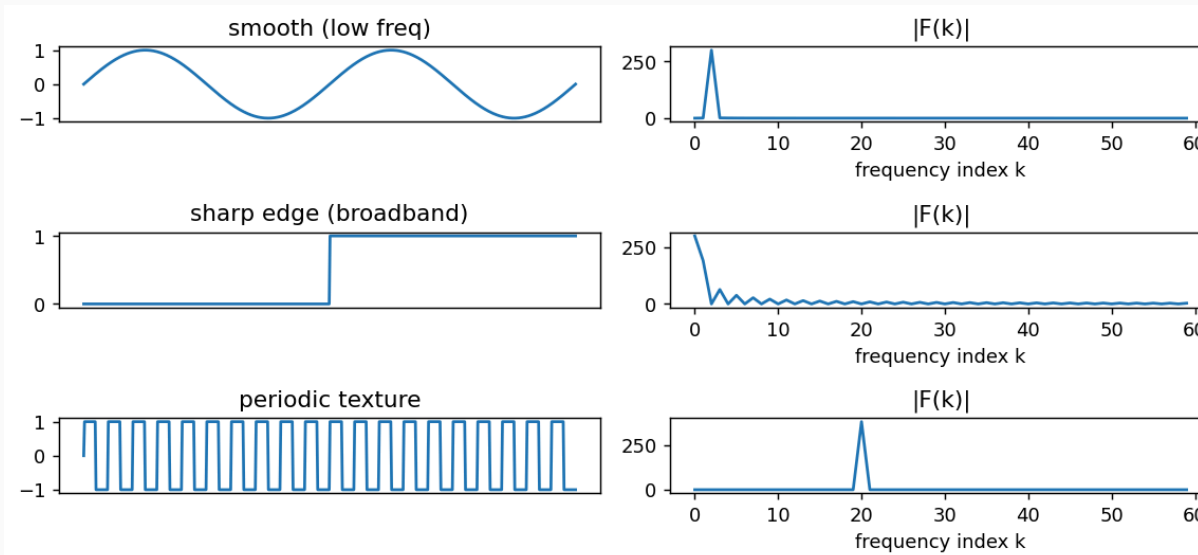


Figure 23: 1D signals (left) and magnitude of their spectra (right).

- Let $\delta(x, y)$ an impulse at the origin. $\delta(0, 0) = 1$ and it is 0 everywhere else.

$$\mathcal{F}\{\delta[n, m]\} = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \delta[n, m] \exp\left(-2\pi j\left(u\frac{n}{N} + v\frac{m}{M}\right)\right) = 1$$

- An impulse contains **all frequencies with equal magnitude and zero phase**
- Impulses are used to probe a system's frequency response

Note: to familiarize yourself with the Fourier transform you should do the computation and verify this result

- pure cosine and sine waves result in pairs of complex frequencies
- $\mathcal{F}\left\{\cos\left(2\pi\left(\frac{u_0n}{N} + \frac{v_0m}{M}\right)\right)\right\} = \frac{1}{2}(\delta[u - u_0, v - v_0] + \delta[u + u_0, v + v_0])$
- $\mathcal{F}\left\{\sin\left(2\pi\left(\frac{u_0n}{N} + \frac{v_0m}{M}\right)\right)\right\} = \frac{1}{2j}(\delta[u - u_0, v - v_0] - \delta[u + u_0, v + v_0])$
- where δ is the impulse signal

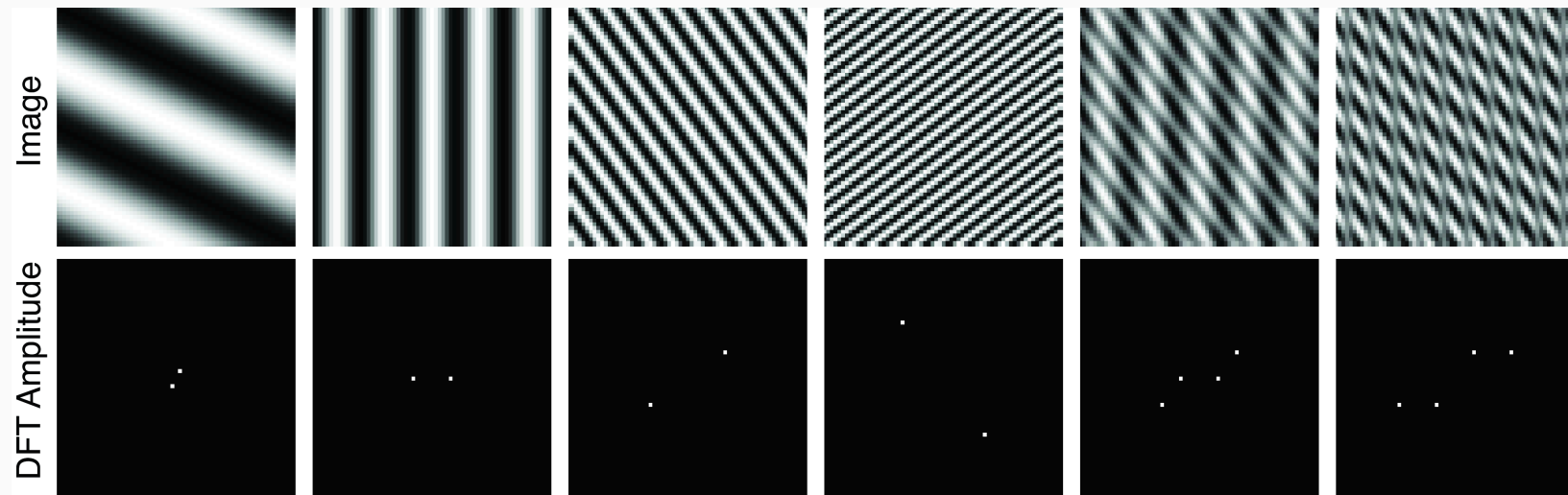


Figure 24: source: visionbook.

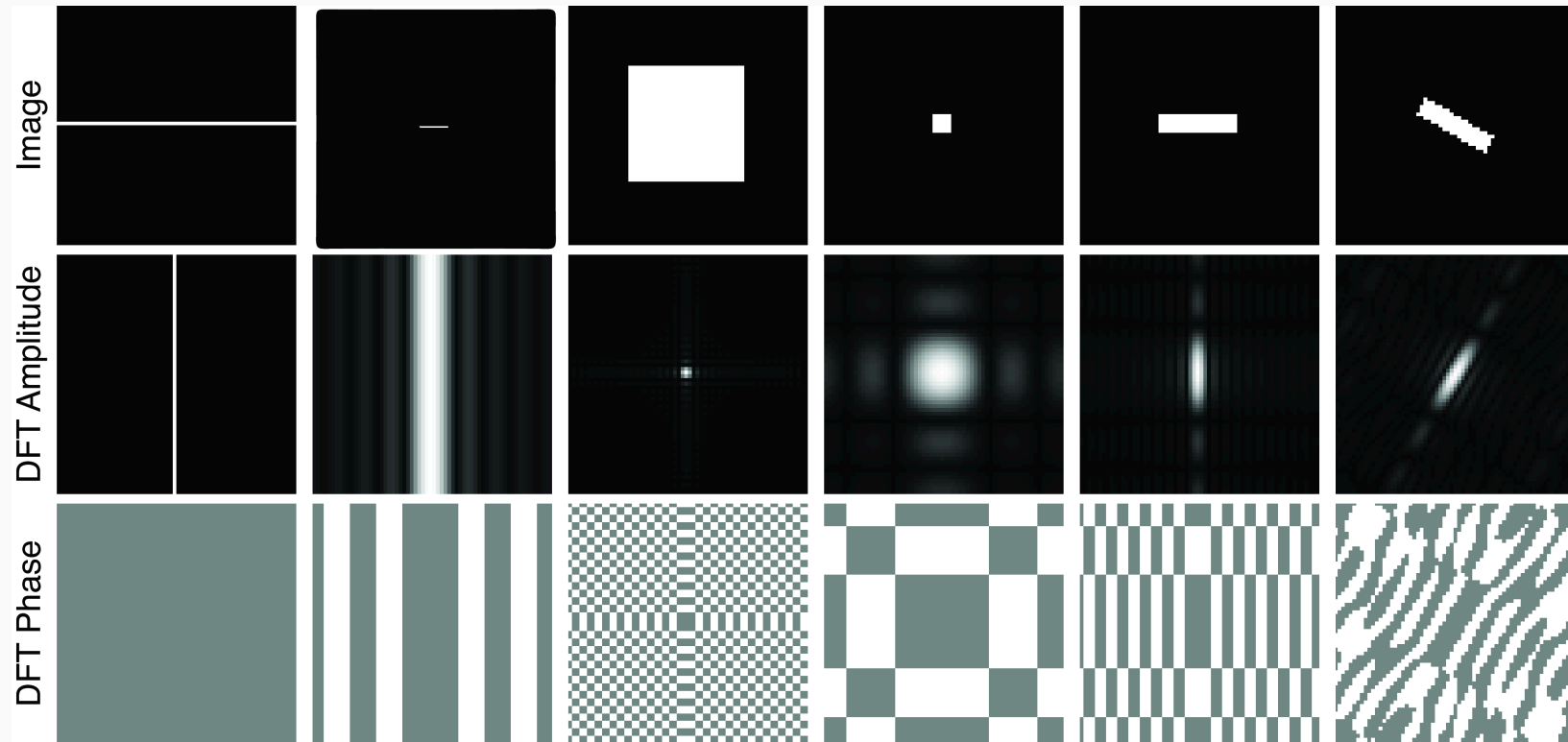


Figure 25: source: visionbook.

Magnitude vs Phase: Which Matters More?

- Classic experiment: swap the magnitude and phase spectra of two images
- Result: the **phase** largely determines perceived image structure; magnitude mostly affects contrast/texture style
- Implication: edges, shapes, object boundaries are encoded mostly in phase relationships across frequencies, important for understanding what filters preserve or destroy



Figure 26: Swapping magnitude and phase between two images: structure follows phase

Magnitude vs Phase

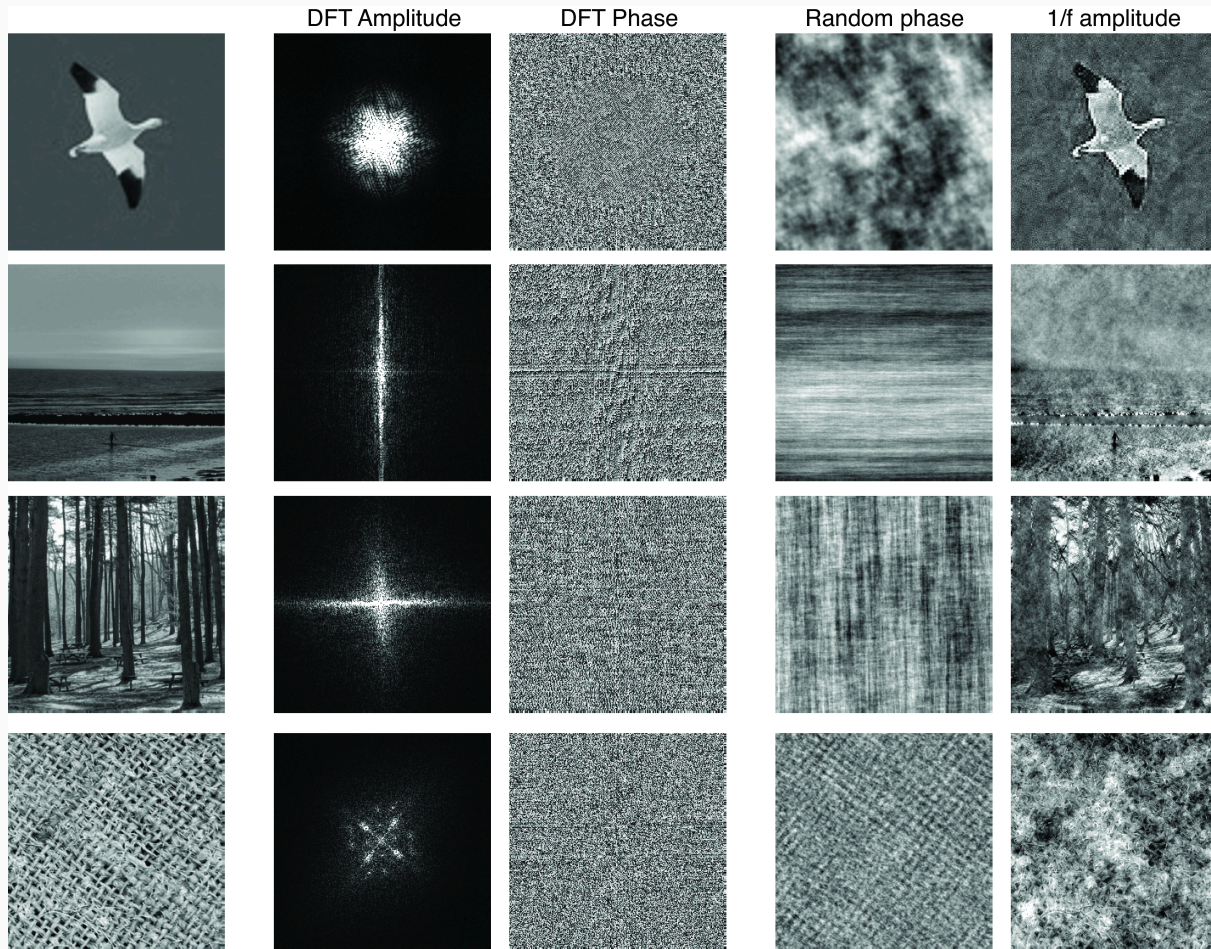


Figure 27: Swapping magnitude and phase between two images: structure follows phase

Properties of the Fourier Transform

- **Linearity**: $\mathcal{F}\{af + bg\} = a\mathcal{F}\{f\} + b\mathcal{F}\{g\}$
 - true for real and complex signals
- For real-valued $l(x)$: $\mathcal{F}(u) = F(-u)^*$ (**conjugate symmetry**)
 - real signals/images have symmetric magnitude spectra (symmetry around the DC term $(0, 0)$)
 - Negative frequencies carry no extra information for real signals, they're the mirror image of positive frequencies
 - This is why numpy provides `rfft`: it only computes half of the frequencies

$$\mathcal{F}\{f * h\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{h\}$$

Convolution in the spatial domain is **pointwise multiplication** in the frequency domain.

consequences:

- Convolution is filtering: for example, a low-pass filter is just a function $H(\xi)$ that is ≈ 1 for low $|\xi|$ and ≈ 0 for high $|\xi|$
- Filter design becomes designing a frequency response curve
- Cascading filters is equivalent to multiplying their frequency responses (consistent with associativity of convolution)

- By the Convolution Theorem: $f * h = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{h\}\}$
- For large kernels, FFT-based convolution ($O(N \log N)$) beats direct convolution ($O(Nk)$) when k is large
- Caveat: DFT assumes circular signals, must zero-pad to avoid wrap-around artifacts when computing **linear** convolution via FFT
- Deep learning frameworks (cuDNN, cuFFT) switch between direct and FFT-based convolution algorithms depending on kernel size

We can fully describe an LSI system in two ways:

- **spatial view**: with the convolution kernel $h(x)$
 - $h(x)$ is the **impulse response**, that is the output of a filter given an impulse $\delta(x)$
- **frequency view**: $H(\xi) = \mathcal{F}\{h\}$ is the **transfer function** (frequency response)
 - how much each frequency is amplified/attenuated and phase-shifted
- h (spatial) and H (frequency) are **Fourier pairs**

- a consequence of the spatial/frequency duality is that LSI systems can be understood as amplifying or attenuating frequencies

Filter families:

- Low-pass filter: passes low frequencies, attenuates high. Blurring effect
- High-pass filter: attenuates low frequencies. Enhances edges/details
- Band-pass filter: isolates a frequency range. Texture/orientation analysis

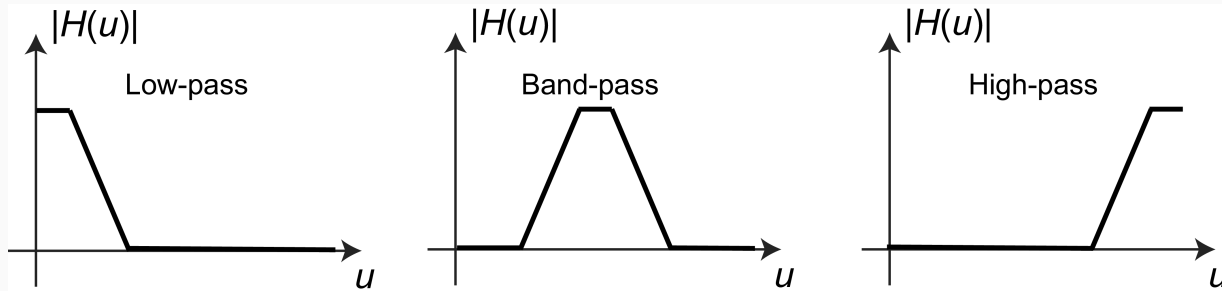


Figure 28: source: visionbook

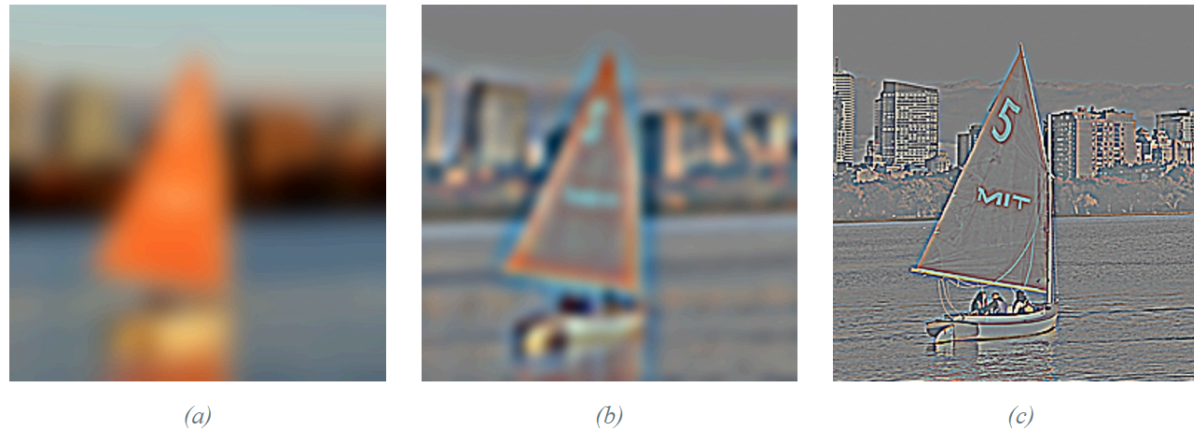


Figure 29: source: visionbook

Applications and Connections with DNNs

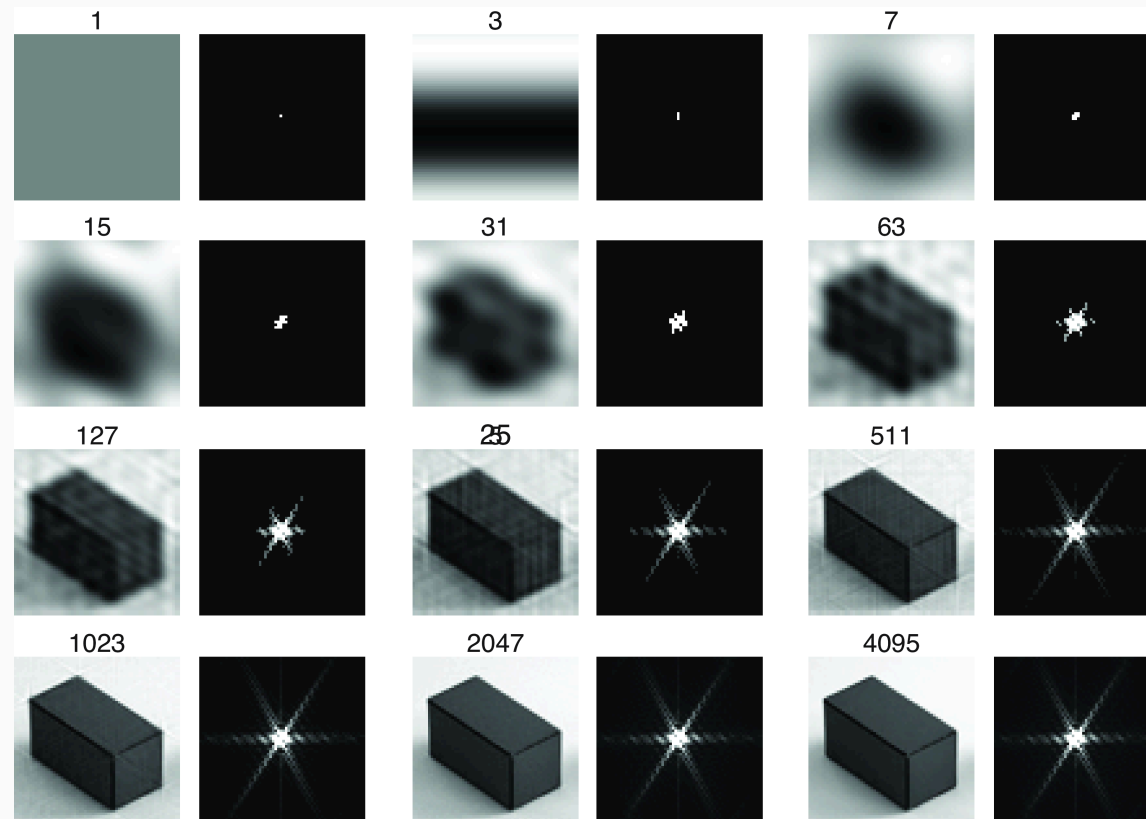


Figure 30: Reconstructing an image from the N Fourier coefficients of the largest amplitude. The right frame shows the location, in the Fourier domain, of the N Fourier coefficients, which when inverted, give the image at the left. Source: visionbook

- JPEG operates on 8×8 blocks via the **Discrete Cosine Transform (DCT)**, a close relative of the DFT (real-valued, even symmetry)
- Quantizes high-frequency DCT coefficients more aggressively (less perceptually important), direct application of frequency-domain reasoning
- Blocking artifacts at low bitrates are a consequence of the underlying algorithm

- **Convolutional layers**: each filter has a frequency response $H(\xi, \eta)$; learned filters often resemble band-pass/oriented filters (cf. Gabor-like filters in early CNN layers)
- **Stacking convolutions**: by the Convolution Theorem, equivalent to multiplying frequency responses, effective filter = product of per-layer responses
- **Nonlinearities (ReLU, etc.)**: break the clean LSI/Fourier picture by introducing new frequencies (harmonics), this is part of why deep nets can model complex frequency content

- Larger receptive field (deeper stacks, larger/dilated kernels) → can represent **lower frequency** (more global) information
- Small kernels stacked → can approximate larger effective kernels (e.g., two $3 \times 3 \approx$ one 5×5 receptive field), but frequency response differs due to nonlinearities in between
- Dilated convolutions: deliberately sample the kernel sparsely, risk of aliasing-like artifacts (gridding artifacts) if not designed carefully

- Many textures can be characterized by their dominant spatial frequencies and orientations (e.g., Fourier/Gabor-based texture descriptors)
- Band-pass filter banks (Gabor filters, wavelets) decompose images into oriented frequency bands, foundation for classical texture analysis and early CNN-like architectures

Conclusions

- **Convolutions** are LSI systems
- **Fourier transform** is a change of basis to an orthogonal sinusoidal basis, revealing frequency structure
- Convolutions are multiplication in the frequency space. We can analyze convolutional filters in the frequency domain to understand their behavior
- Signal processing tool help design and explain modern CNNs (dilated convs, pooling, frequency-domain filtering)

- **Lecture 4 (Lab)**: hands-on FFT on 1D signals
- **Lecture 5**: spatial/temporal filters as specific convolution kernels, gradients (derivative filters)
- **Lecture 6**: sampling, aliasing, image pyramids, and multi-resolution representations

- visionbook Ch. 15 and 16
- try to do yourself some of the computations in the book