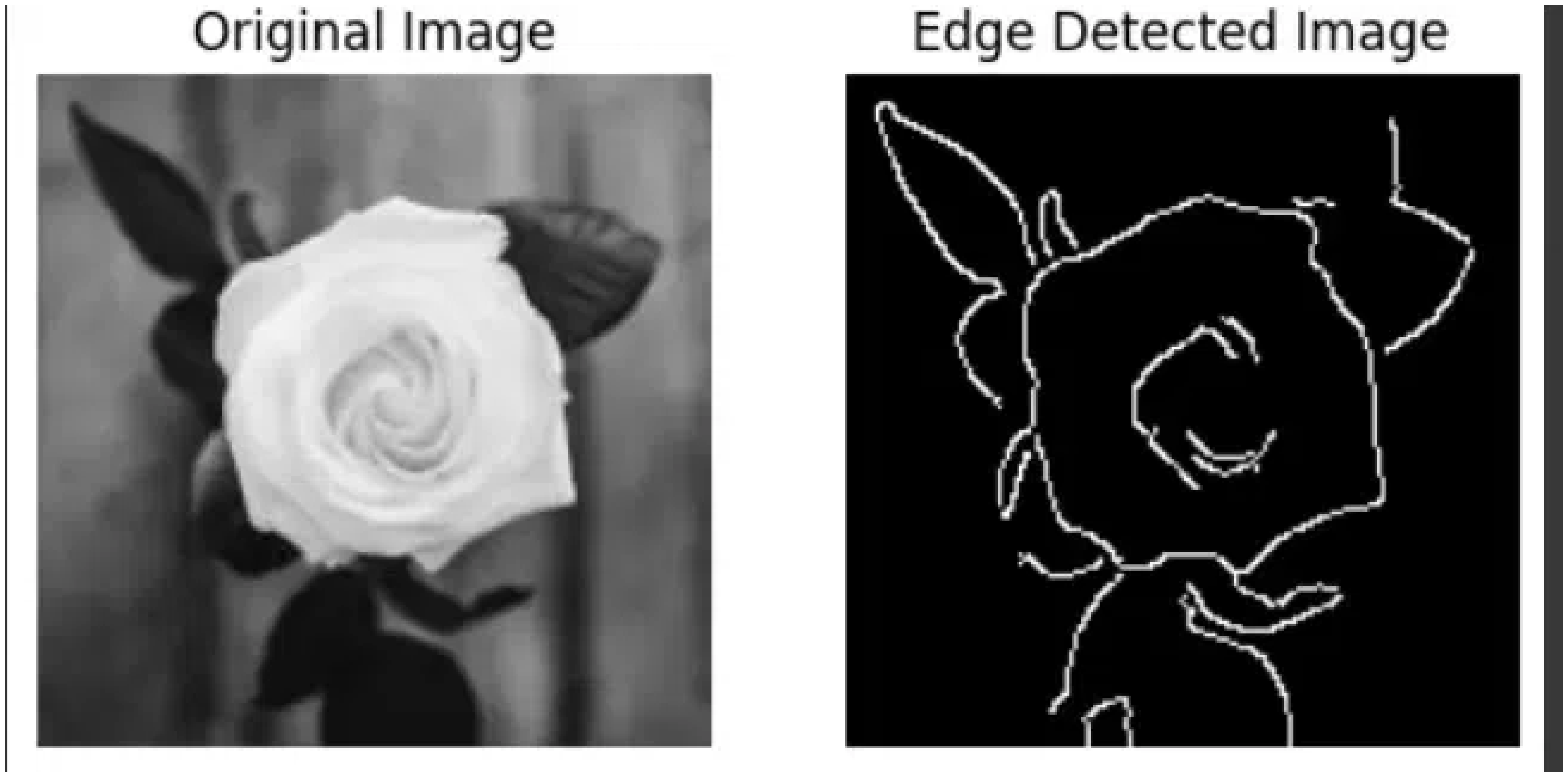
Image Processing 🧠

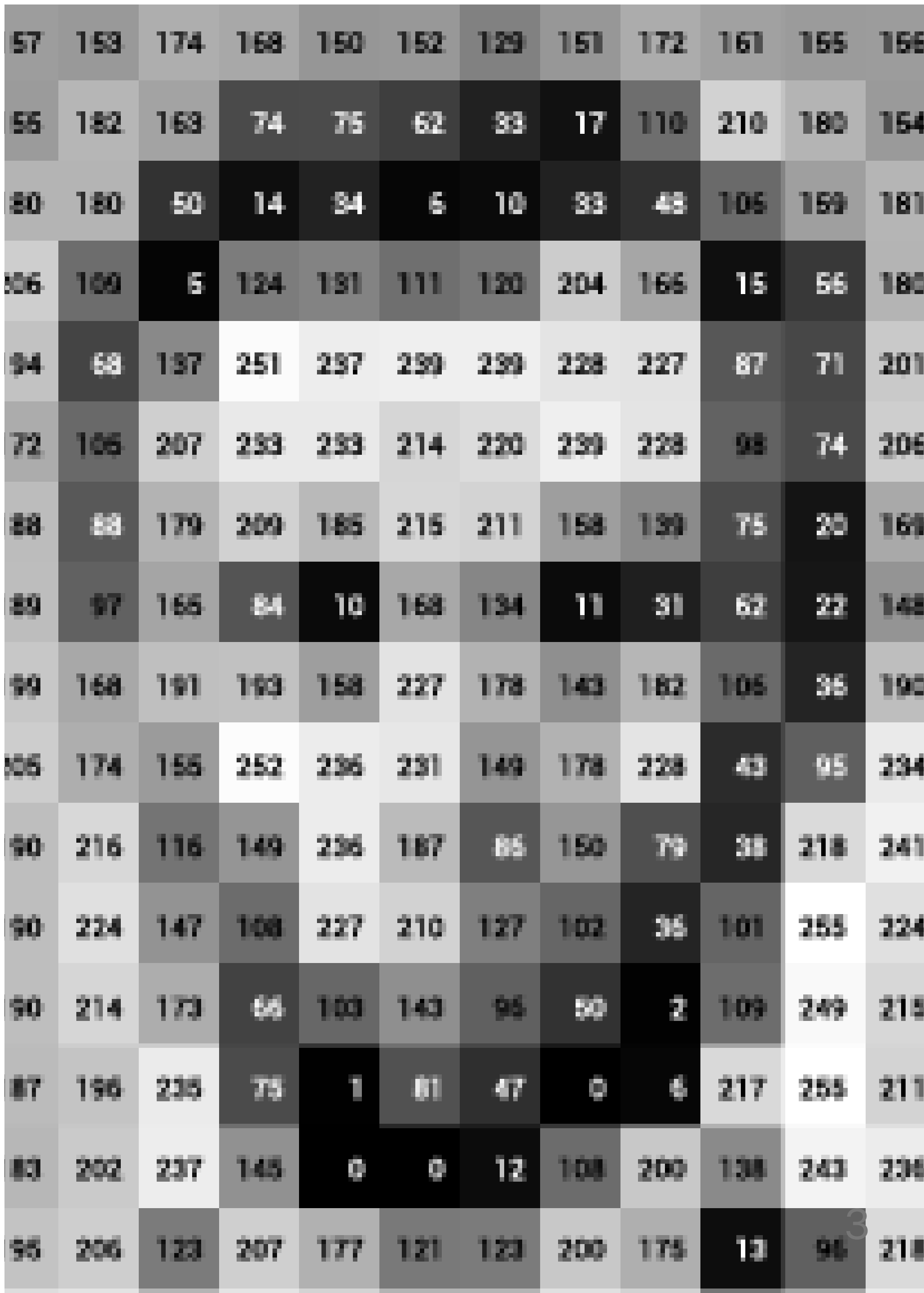
Image Processing Fundamentals: Edge Detection 🖼️



Basic Image Processing 📸

- Image = 2D matrix of pixel values
- Each pixel represents brightness/color
- Example grayscale image:

$$X = \begin{bmatrix} 10 & 10 & 80 & 10 & 10 & 10 \\ 10 & 10 & 80 & 10 & 10 & 10 \\ 10 & 10 & 80 & 10 & 10 & 10 \\ 10 & 10 & 80 & 10 & 10 & 10 \\ 10 & 10 & 80 & 10 & 10 & 10 \\ 10 & 10 & 80 & 10 & 10 & 10 \end{bmatrix}$$



Convolution: Spatial Domain 🔄

- Slide kernel over image
- Multiply and sum values
- Example kernel (vertical edge detection):
- Demo

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

Convolution is Complicated 😬

Example:

Suppose we have an image X and a kernel K as follows:

$$X = \begin{bmatrix} X_1 & X_2 & X_3 & X_4 & X_5 & X_6 \end{bmatrix}$$
$$K = \begin{bmatrix} K_1 & K_2 & K_3 \end{bmatrix}$$

The convolution is given by

$$X * K = \sum_{i=1}^{6} X_i K_{7-i}$$

Or equivalently,

$$X * K = \begin{bmatrix} X_1K_3 + X_2K_2 + X_3K_1 & X_2K_3 + X_3K_2 + X_4K_1 & X_3K_3 + X_4K_2 + X_5K_1 & X_4K_3 + X_5K_2 + X_6K_1 \end{bmatrix}$$

Let's make it simpler using the convolution theorem!

What is the convolution theorem?

Suppose two functions f and g and their Fourier transforms F and G. Then,

$$\underbrace{(f * g)}_{\text{convolution}} \leftrightarrow \underbrace{(F \cdot G)}_{\text{multiplication}}$$

The Fourier transform is a one-to-one mapping between f and F (and g and G).

But what is the Fourier transform 🙃?

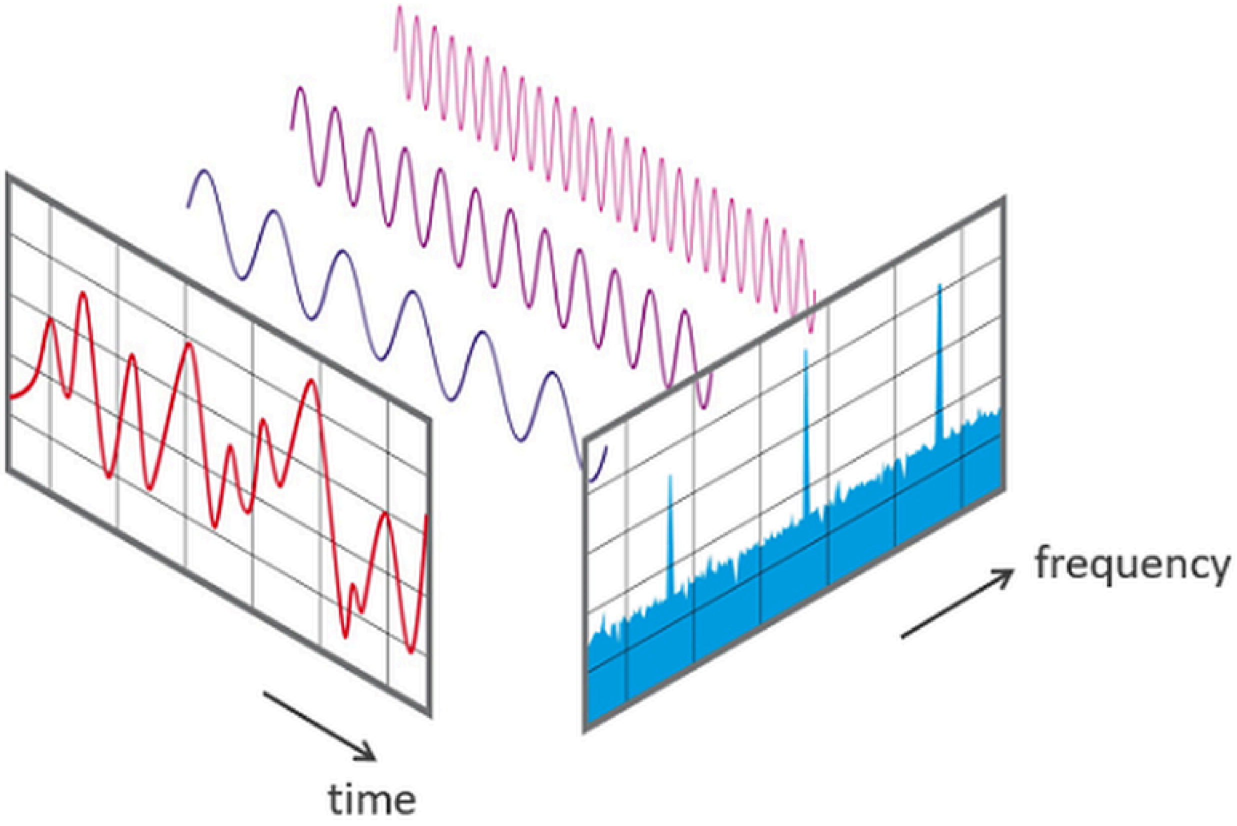
Fourier Transform: The Basics 🌊

Transform a signal from:

- Time/Space domain ➡️ Frequency domain

Key Concept:

- Any signal can be decomposed into sum of sine and cosine waves
- Each wave has specific frequency and amplitude

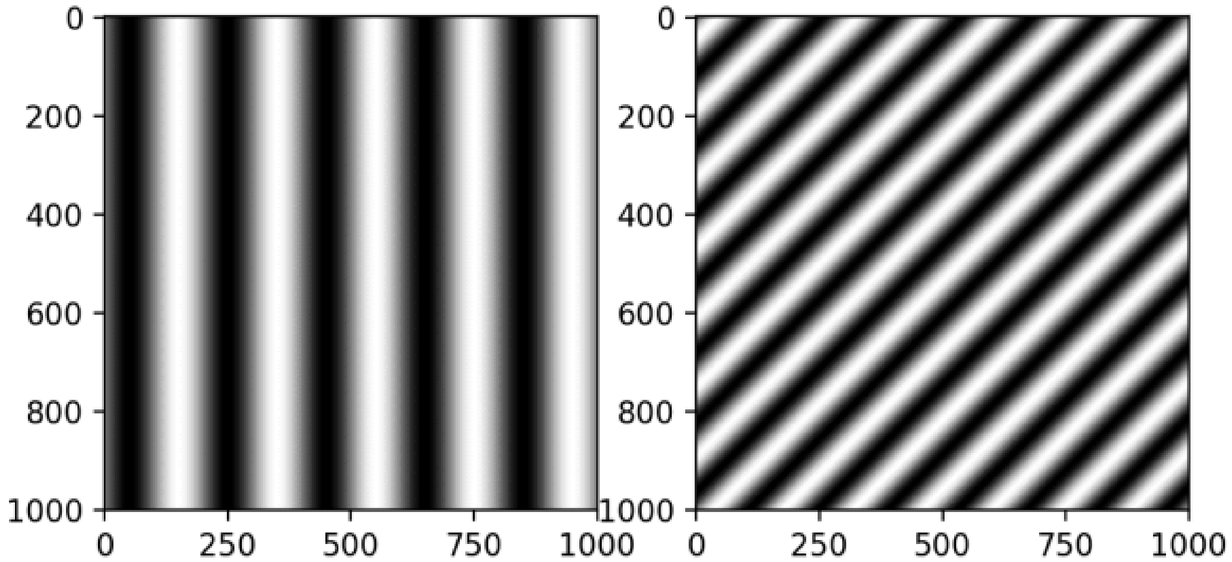


2D Fourier Transform 📊

2D Fourier Transform decomposes image into sum of *2D* waves.

$$\mathcal{F}(X)[h,w] = \sum_{k=0}^{H-1}\sum_{\ell=0}^{W-1} X[k,\ell]\cdot \underbrace{e^{-2\pi i\left(\frac{hk}{H}+\frac{w\ell}{W}\right)}}_{2D\text{ wave}}$$

For image X with size $H \times W$.



Edge Detection in Frequency Domain 🔍

Original Image ➡️ Fourier Transform ➡️ Apply Filter ➡️ Inverse Transform

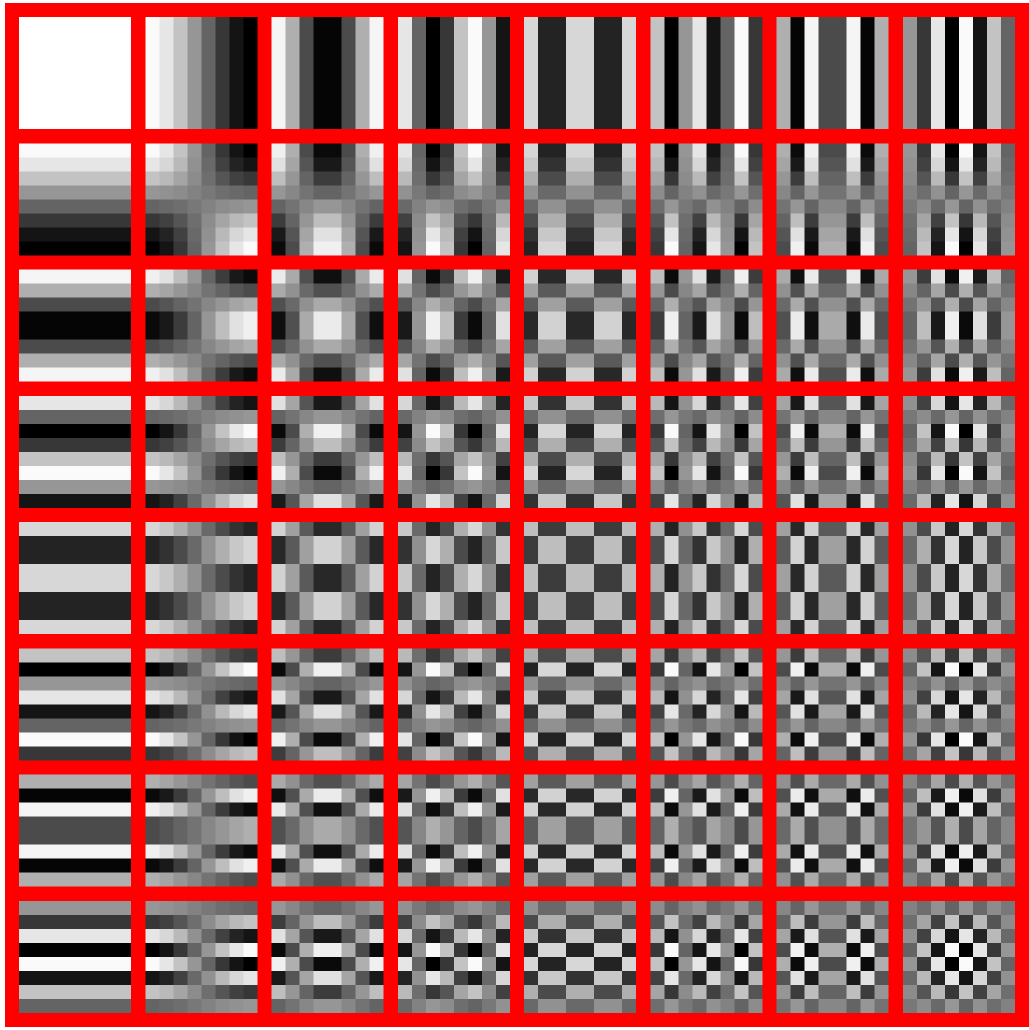
```
# Convert to frequency domain
FX = np.fft.fft2(img_gray) # Image
FK = np.fft.fft2(kernel_padded) # Kernel

# Multiply
filtered = FX * FK

# Convert back
result = np.real(np.fft.ifft2(filtered))
```

JPEG Compression 📸

1. Divide image into 8x8 blocks
2. Apply Discrete Cosine Transform (similar to Fourier)
3. Quantize frequencies
   - Keep low frequencies
   - Discard high frequencies
4. Encode efficiently

Benefits:

- Smaller file size
- Maintains visual quality
- Exploits human visual perception



Coding Exercise

Coding Exercise