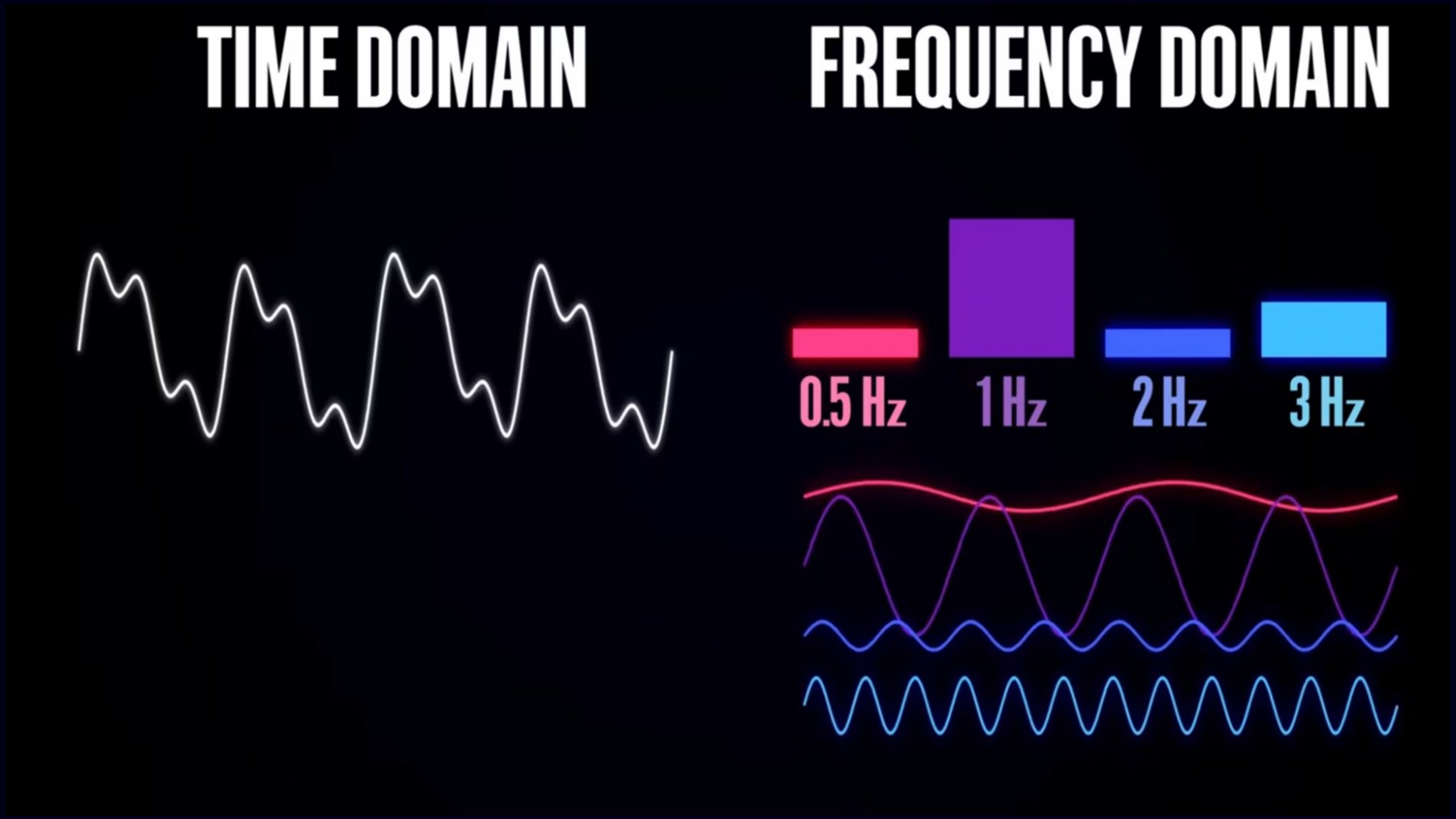
Transform Based Compression

By Mohammad Amin Farahani (CS-Astronaut)

All the codes available on:
https://github.com/CS-Astronaut/Transform-Based-Image-Processing

Time Domain VS Frequency Domain

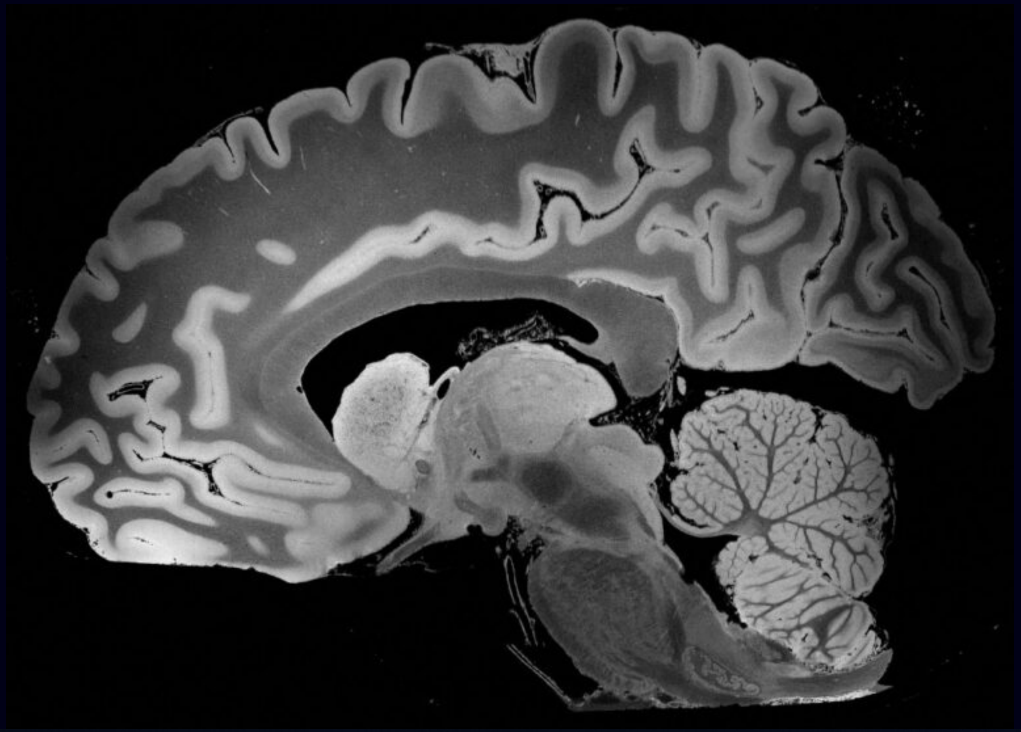


Time Domain:

- Represents signal as it varies over time (or space for images)
- Each point shows a direct value (e.g., pixel intensity)
- Good for visualization but **hard to analyze patterns**

Frequency Domain:

- Represents how **often signal patterns repeat** (frequencies)
- Highlights **periodic structures and correlations**
- Makes **compression, denoising, and filtering** more efficient



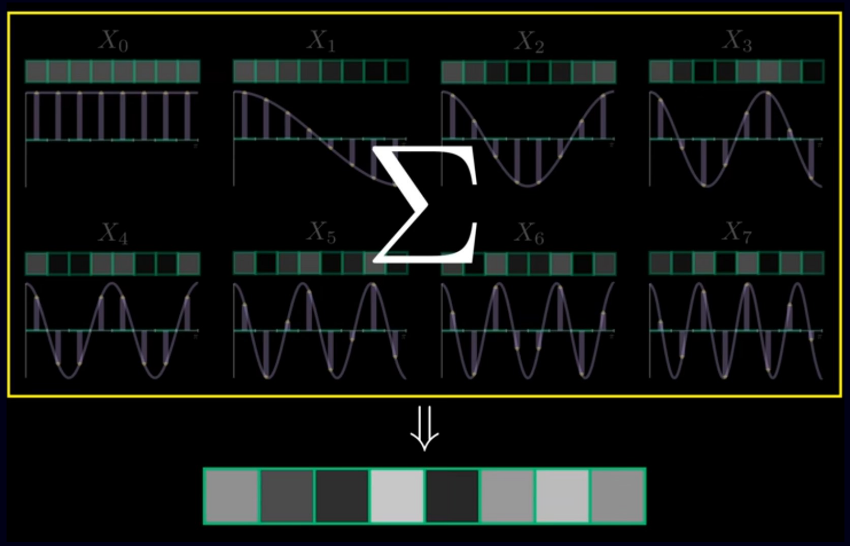
Transform-Based Image Compression

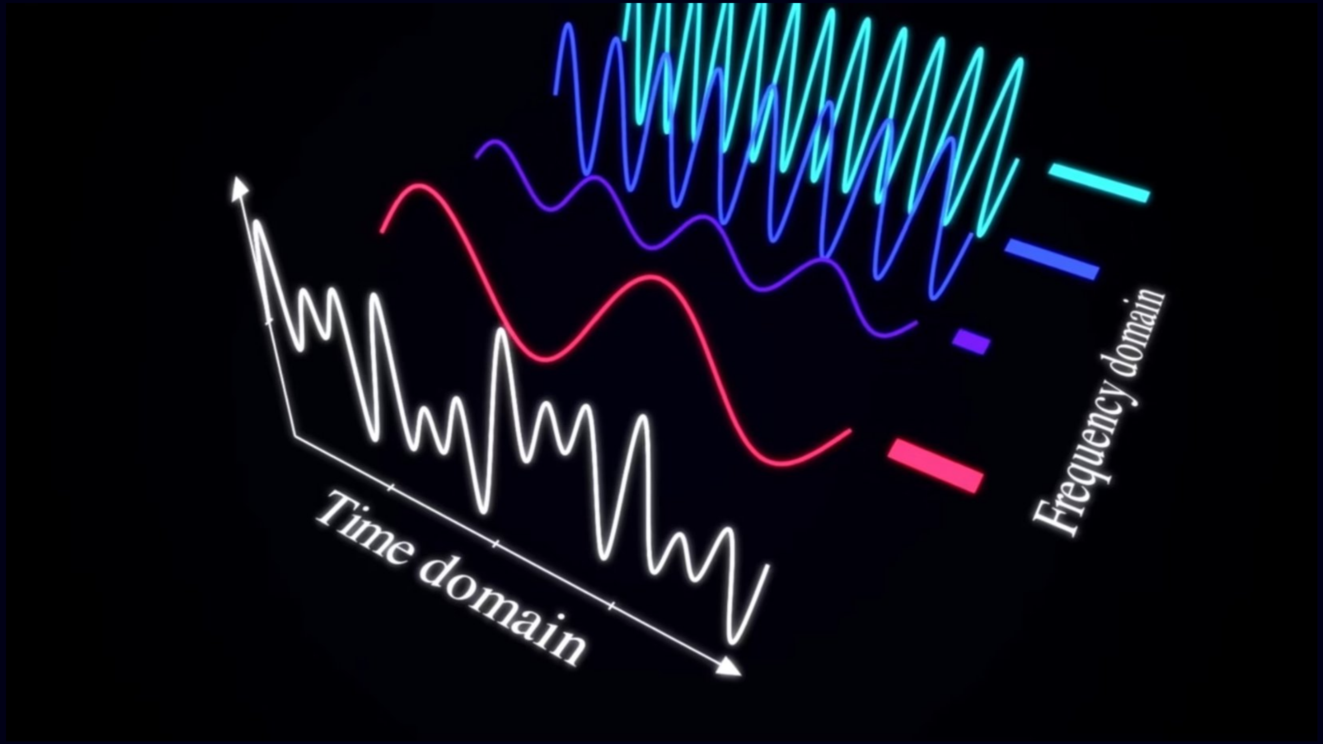
Images are naturally stored in the spatial (pixel) domain.

Compression in the spatial domain is often inefficient due to pixel redundancy.

Allows thresholding, quantization, and efficient encoding.

Enables higher compression ratios with better quality retention.

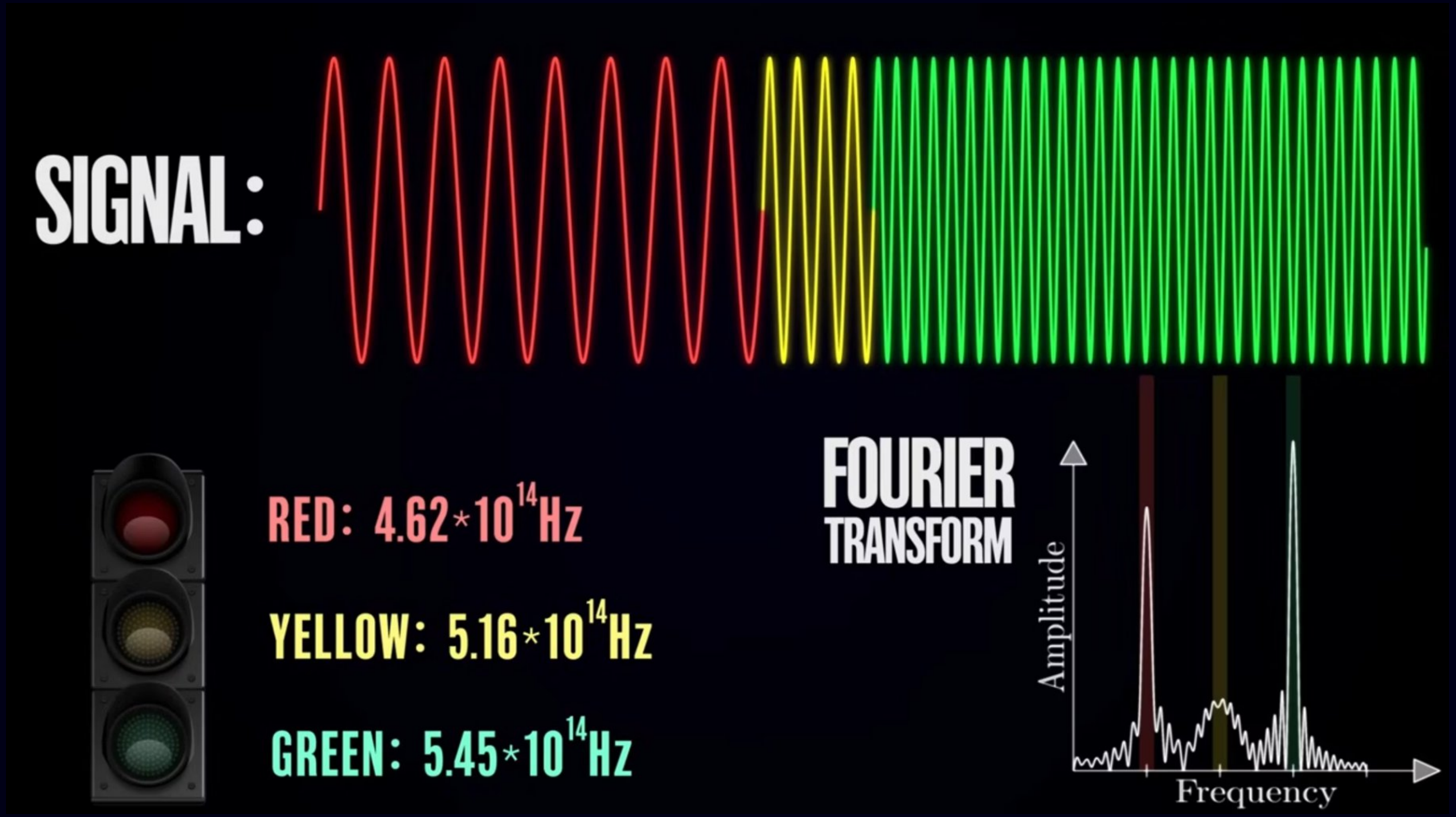


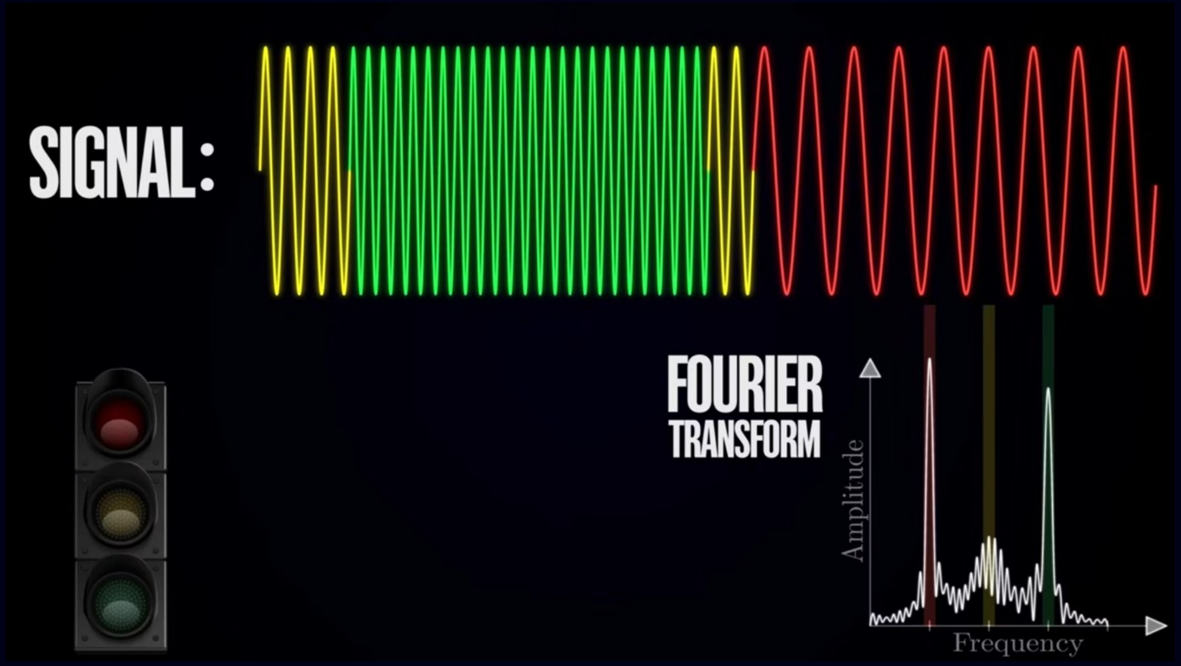


We use the Fourier Transform to break a signal into its basic frequency components

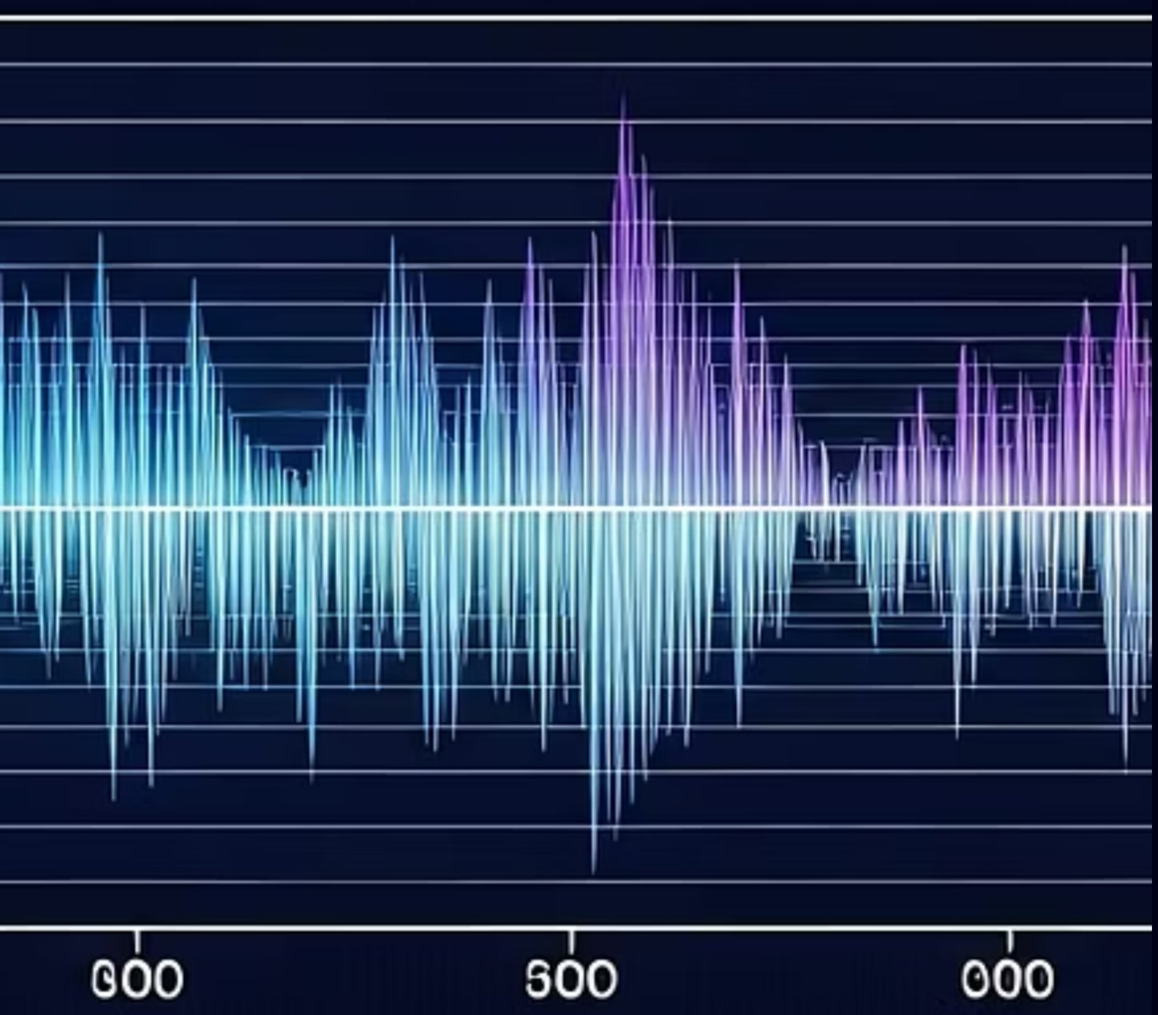
Time Domain > Frequency Domain

an example





so after Fourier transform we can not tell when a frequency occurred we just know a frequency exist



The Fourier Transform: Strengths and Limitations

Frequency Information

Provides frequency information of a signal, representing frequencies and their magnitude

Time Limitation

Does not tell us when in time the frequencies exist

Ideal Application

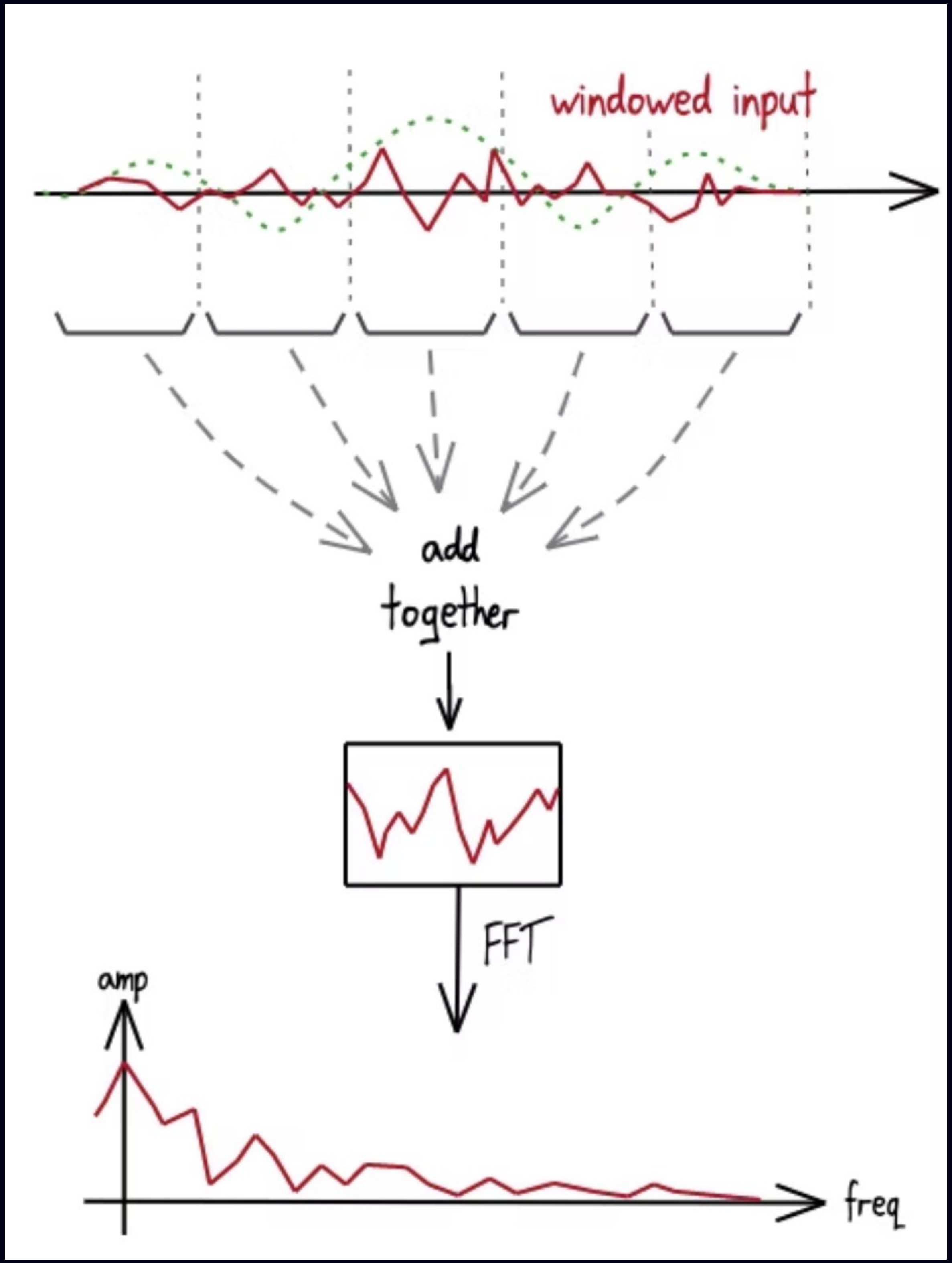
Best for stationary signals where frequency content doesn't change over time

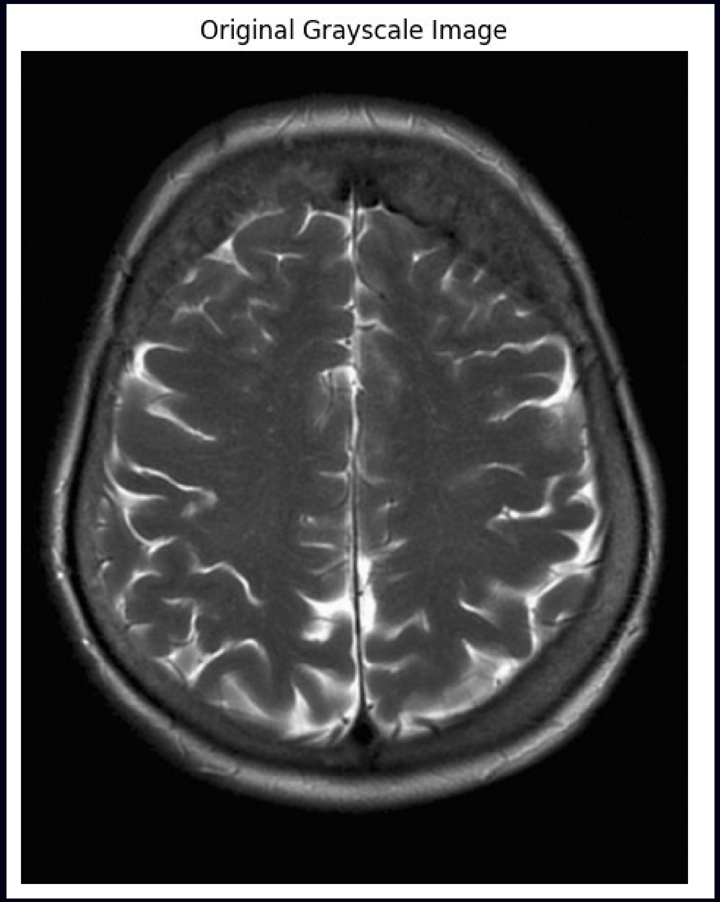
The Fast Fourier Transform (FFT)

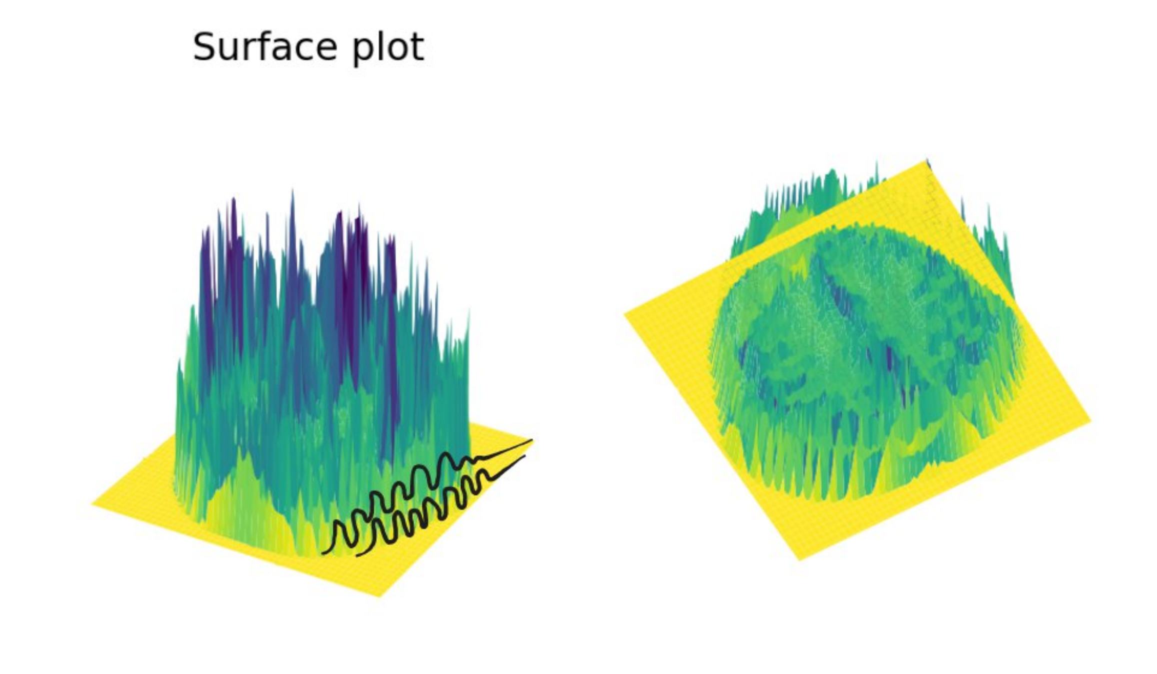
The FFT was developed to overcome the poor time resolution of the Fourier Transform by providing a time-frequency representation of the signal.

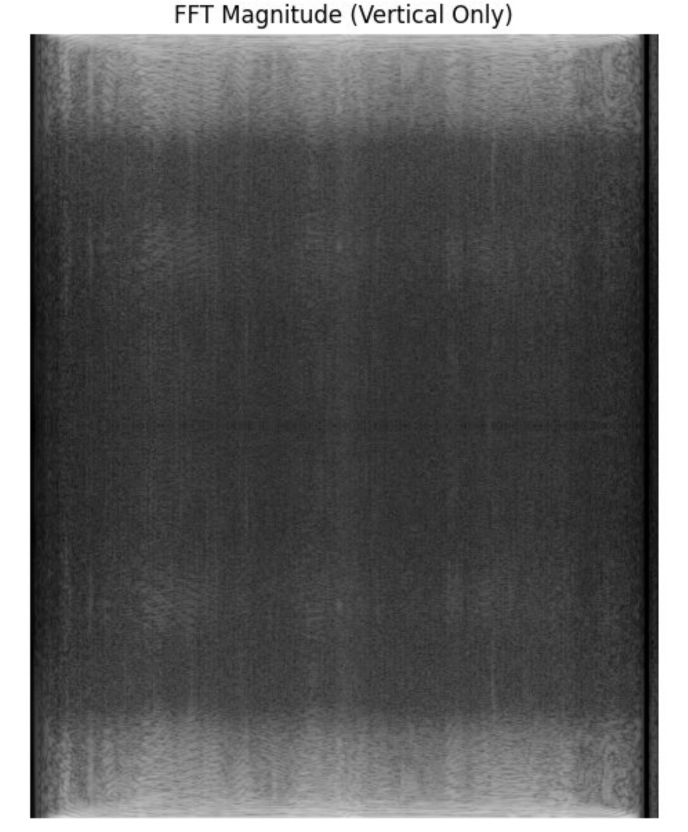
It works by assuming portions of a non-stationary signal are stationary, then taking a Fourier Transform of each stationary portion and adding them up.

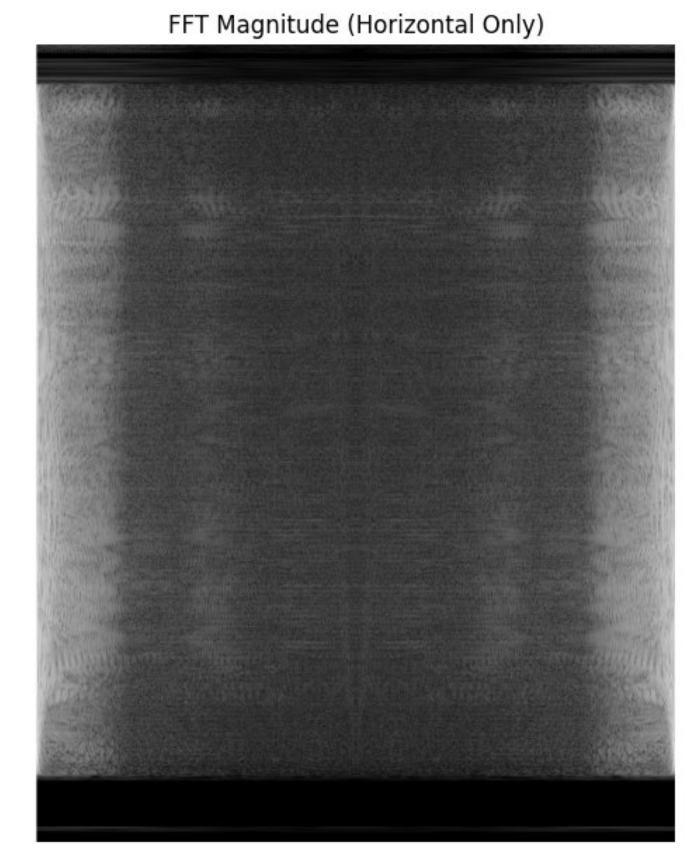
A window function of fixed length moves along the signal from start to end, taking a Fourier Transform at each stationary section.



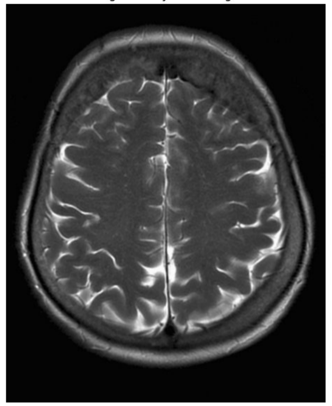

Original Grayscale Image


Surface plot

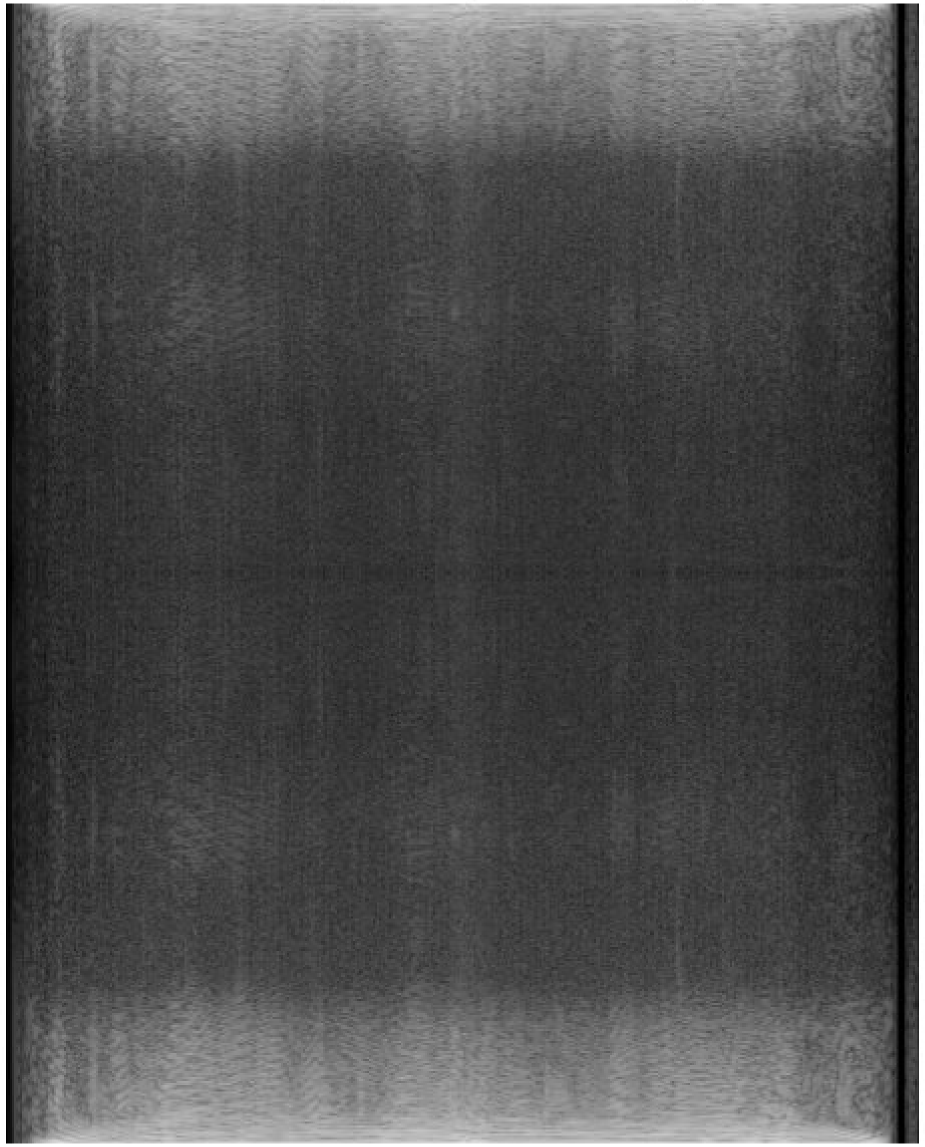

FFT Magnitude (Vertical Only)


FFT Magnitude (Horizontal Only)

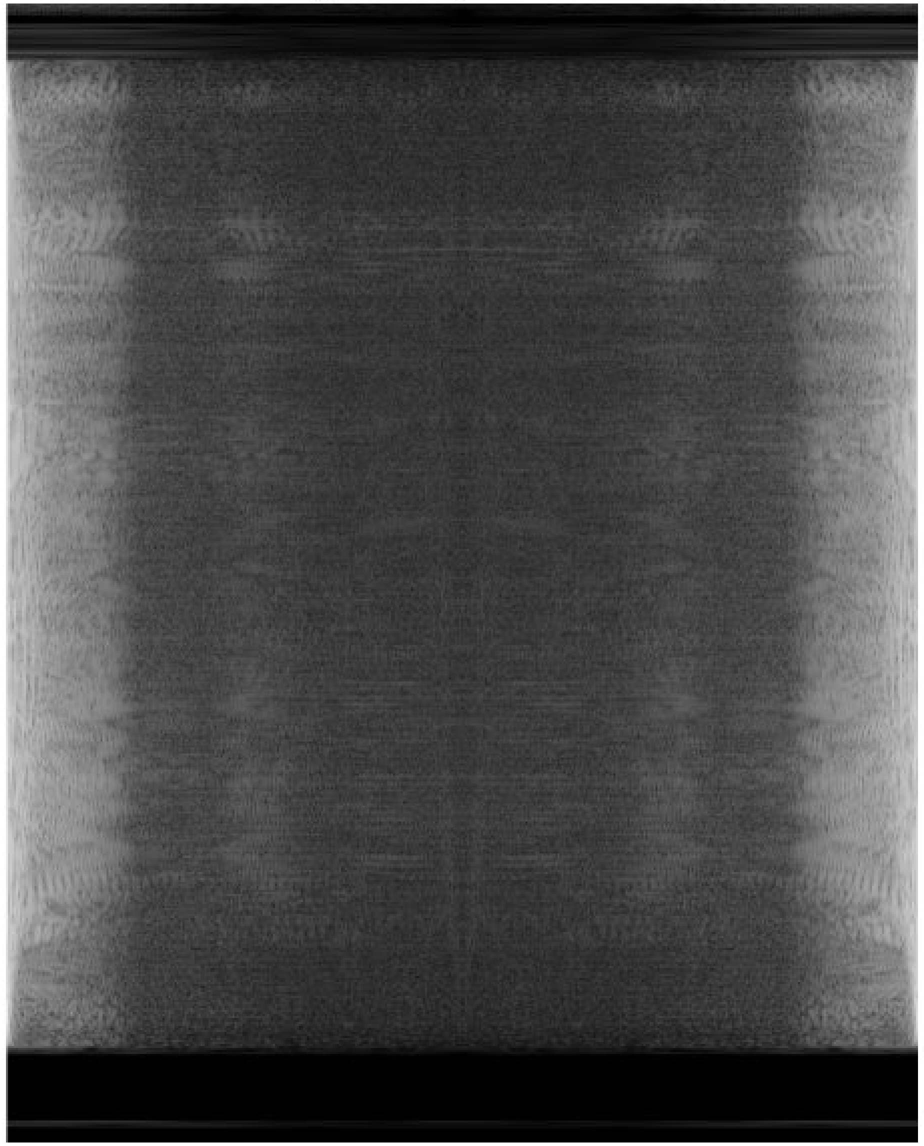
Original Grayscale Image



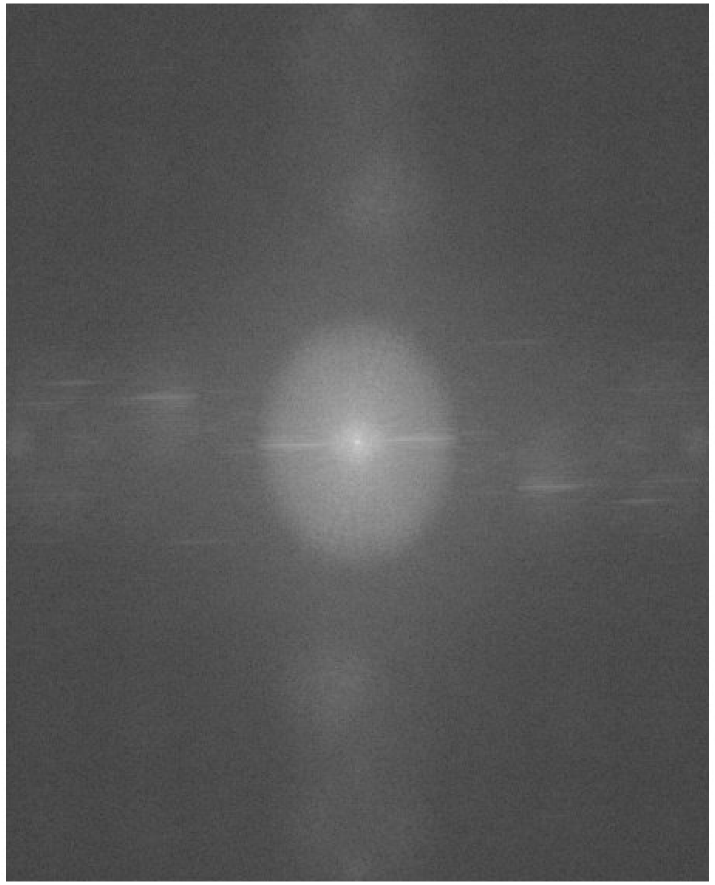
FFT Magnitude (Vertical Only)



FFT Magnitude (Horizontal Only)



FFT Magnitude (2D FFT)



FFT-Based Image Compression

Original Image

Start with the original image in pixel (spatial) space

Compute 2D FFT

Obtain the frequency domain representation

Threshold Coefficients

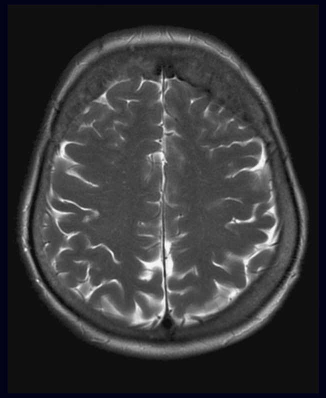
Keep only the largest (e.g., top 1-2%) coefficients

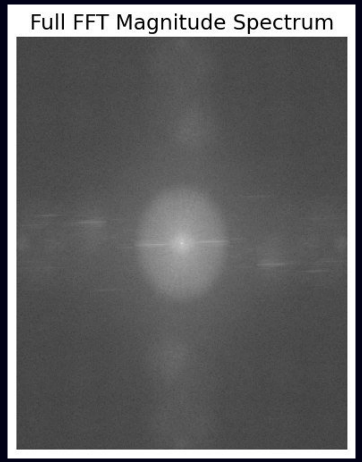
Store Significant Data

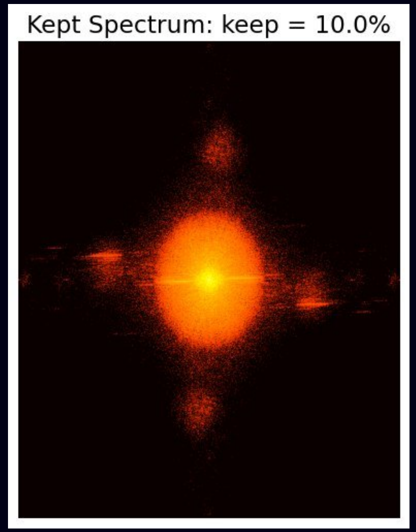
Store only the significant coefficients and their positions

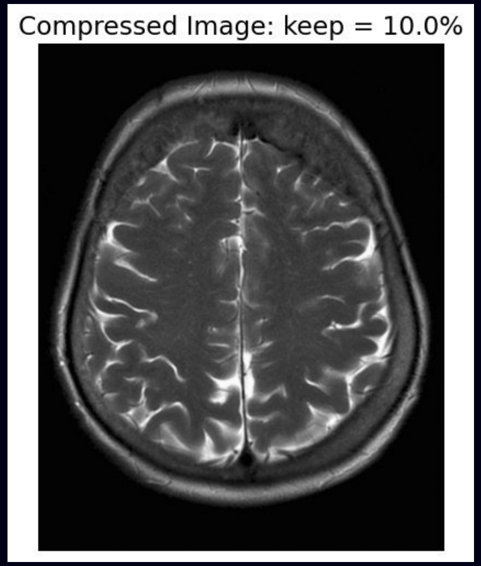
Reconstruct Image

Apply inverse 2D FFT to reconstruct the image

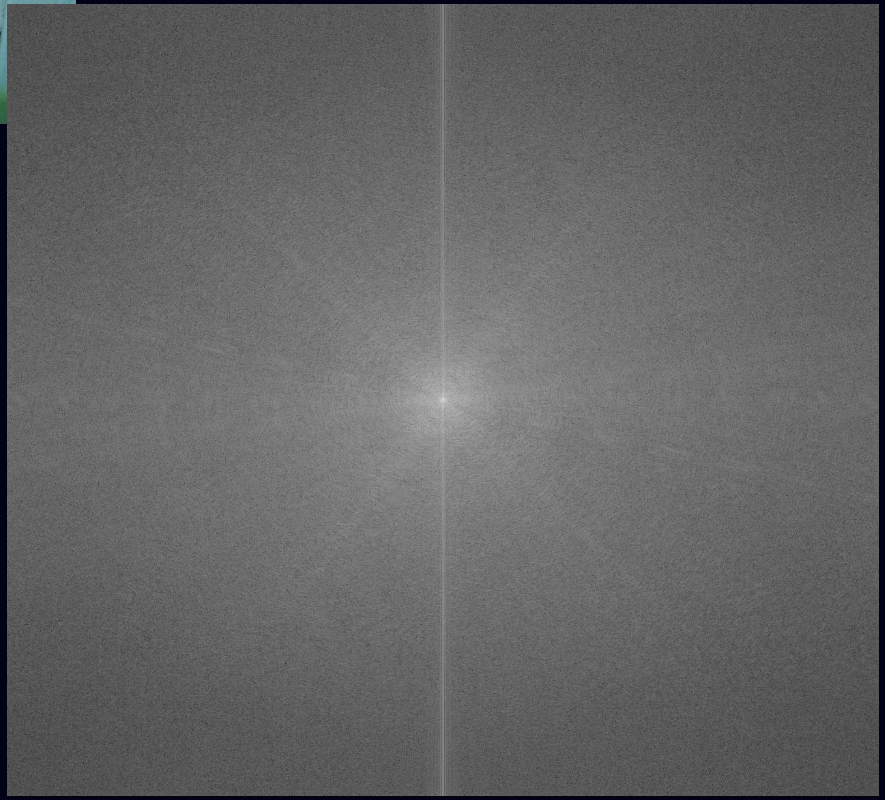




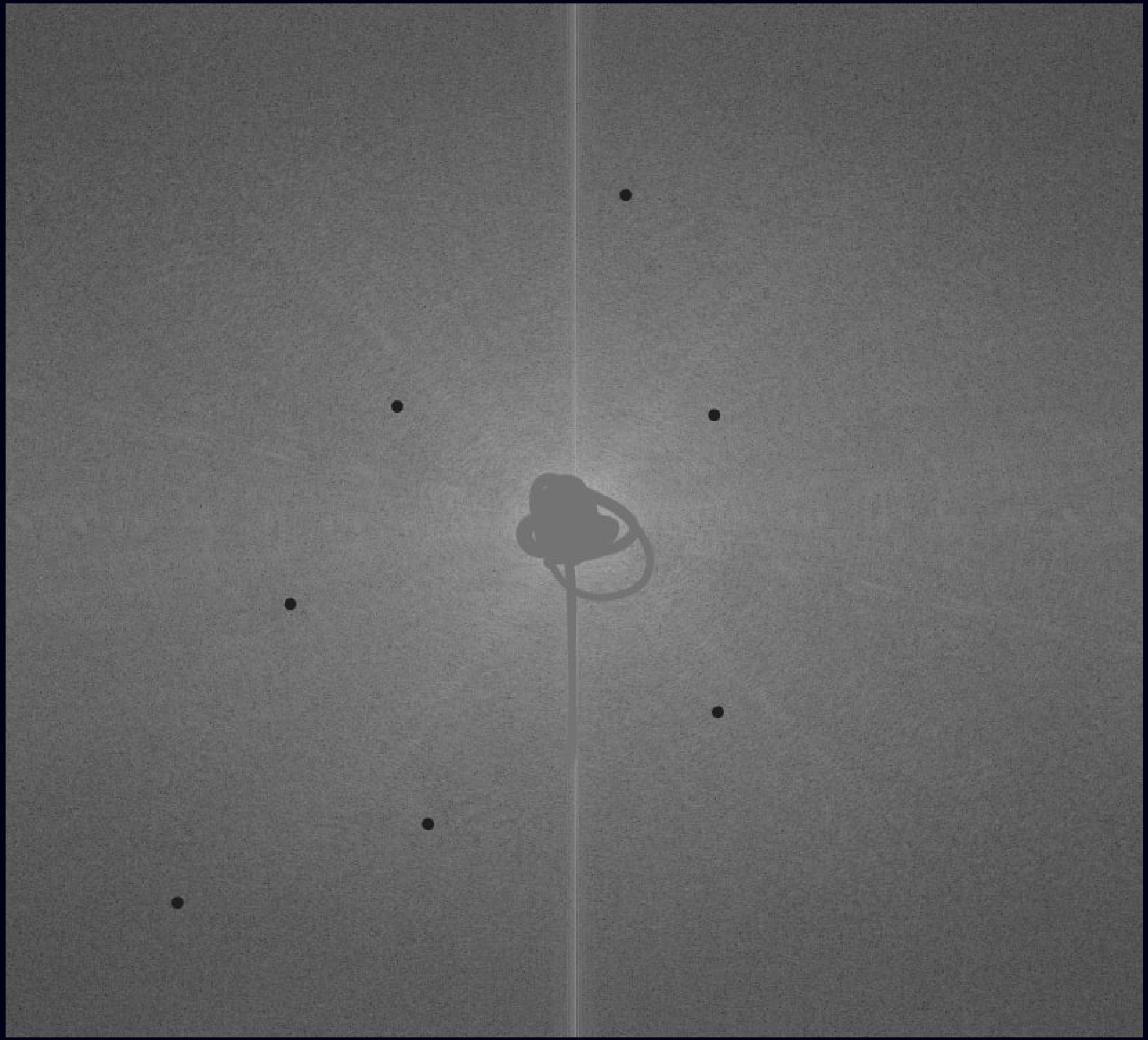




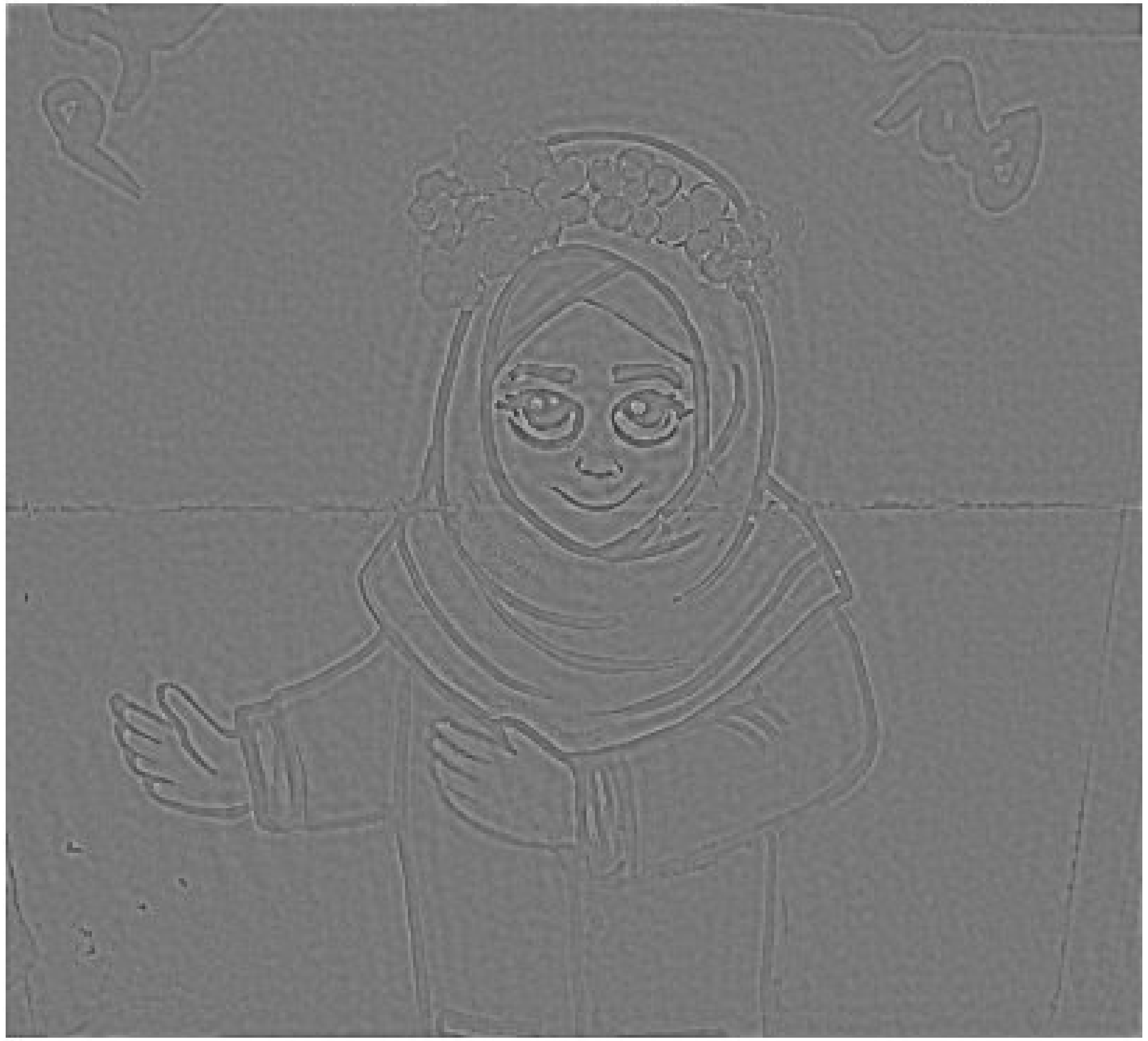
<CODE TIME />



Reconstructed Image (Magnitude + Phase)



Reconstructed Image (Edited Magnitude + Original Phase)



Discrete Wavelet Transform (DWT): Fundamentals

Time-Frequency Resolution

Provides "time-frequency joint representation" identifying what frequencies exist and when they occur

Wavelets as Basis

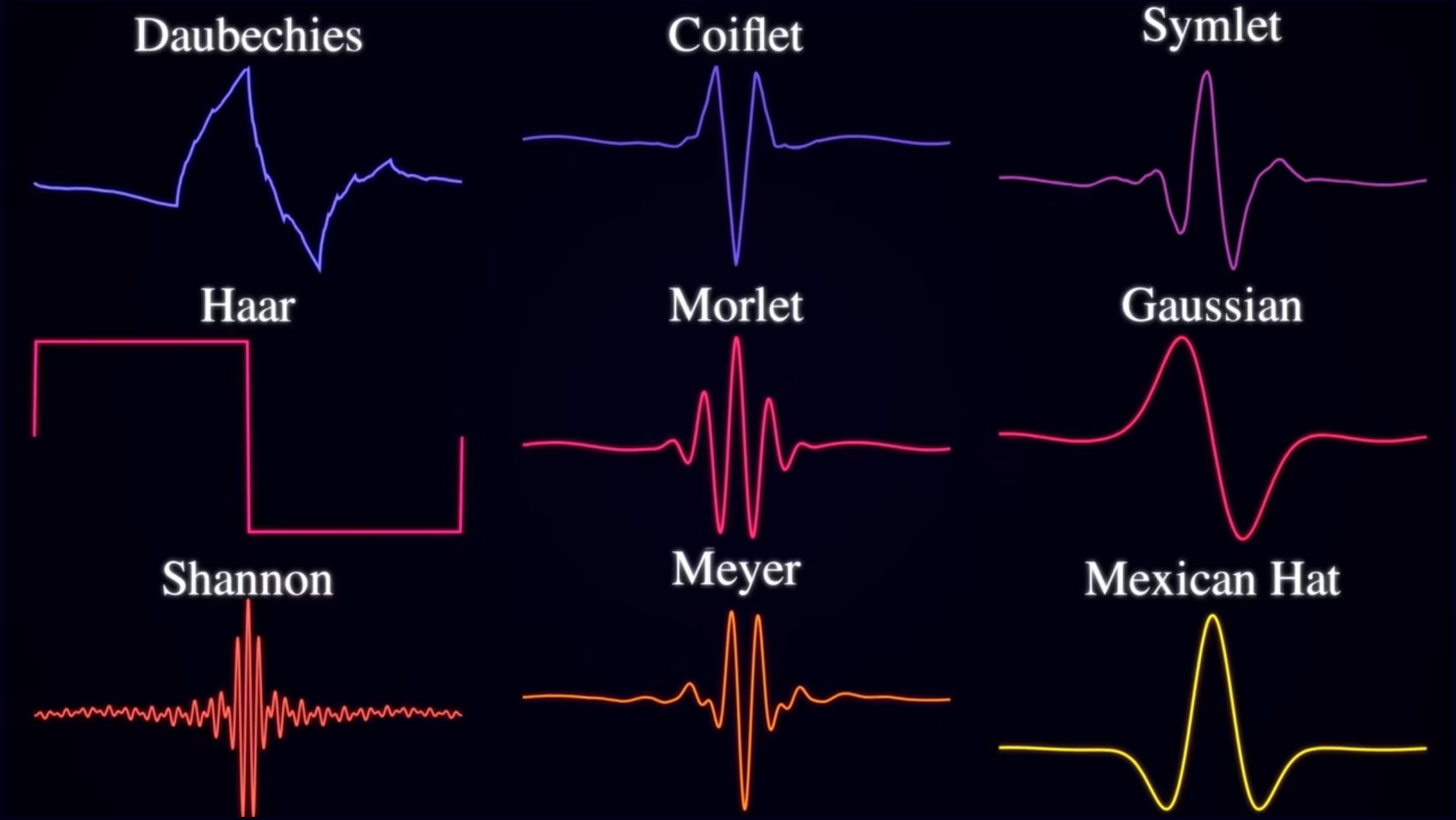
Uses bounded "wavelets" instead of infinite sinusoidal waves, enabling localization in time

Mother & Daughter Wavelets

Generated through scaling (affecting frequency) and translation (shifting along time)

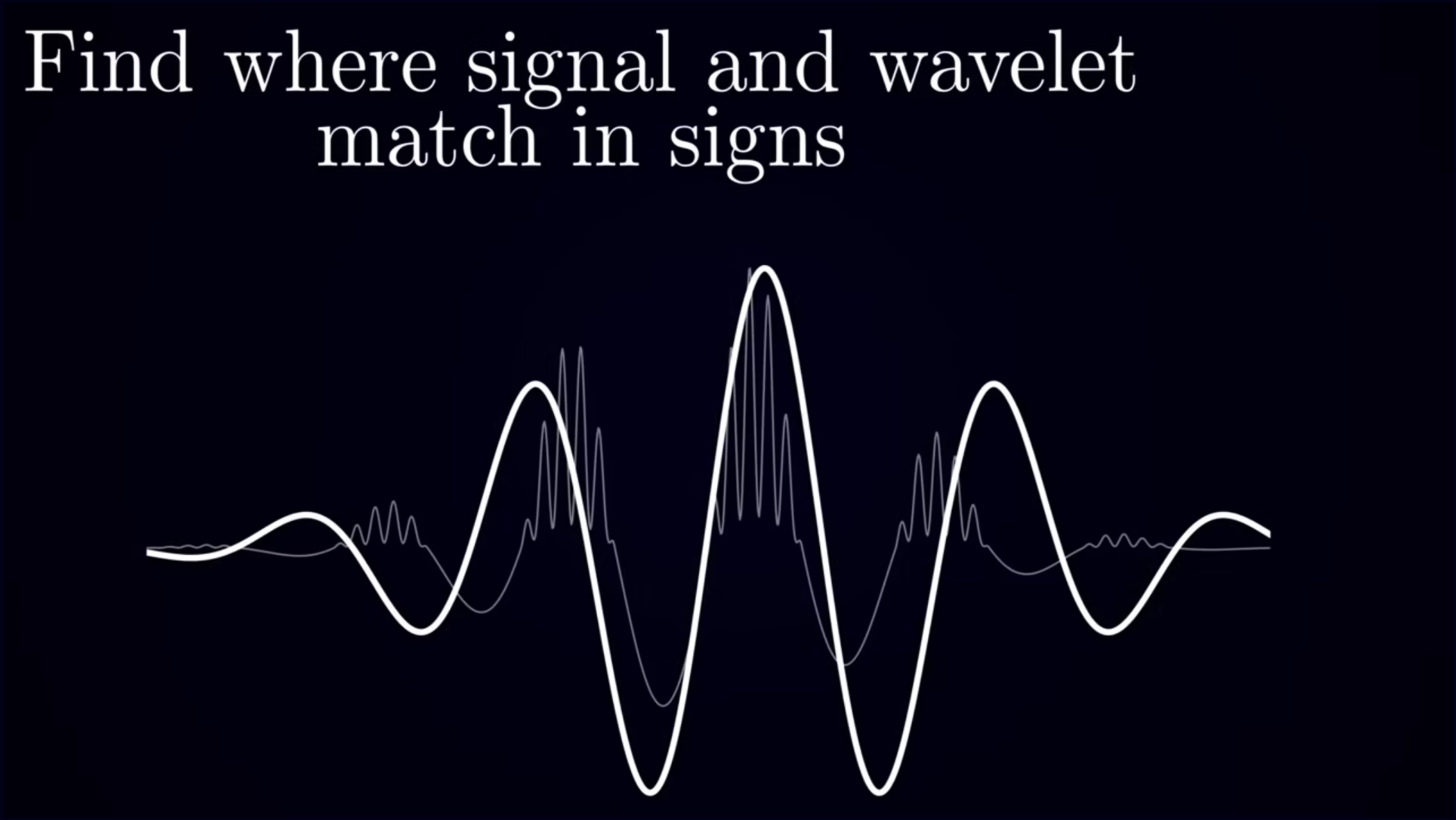
Unlike Fourier, wavelets can "zoom in and out" of signals to extract patterns at different scales

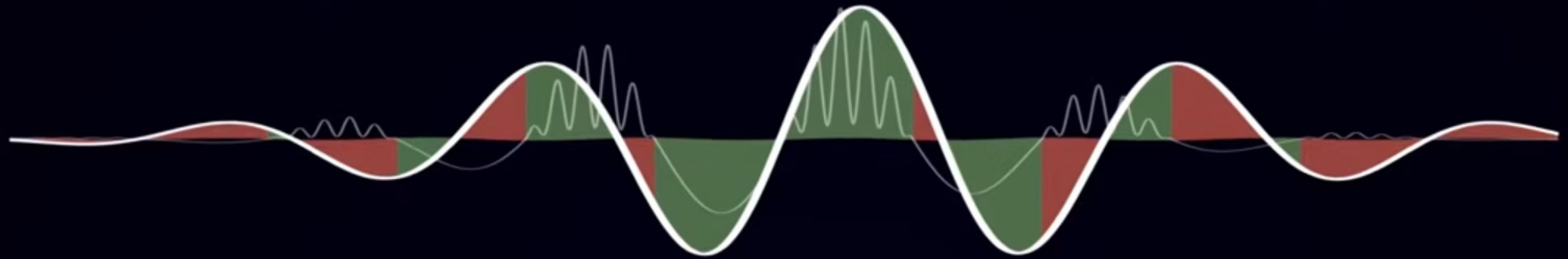
How DWT Works?



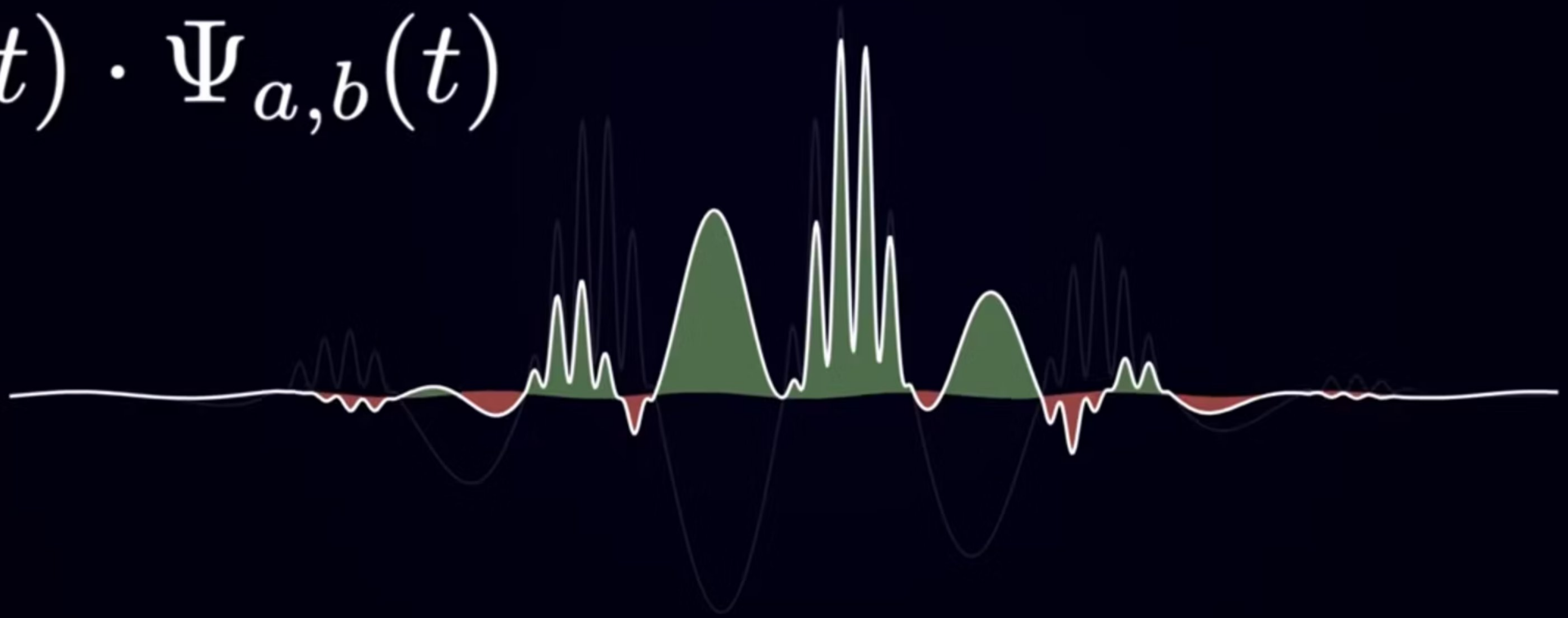
we have different types of wavelets…

After Converting Picture to Spatial Domain we:

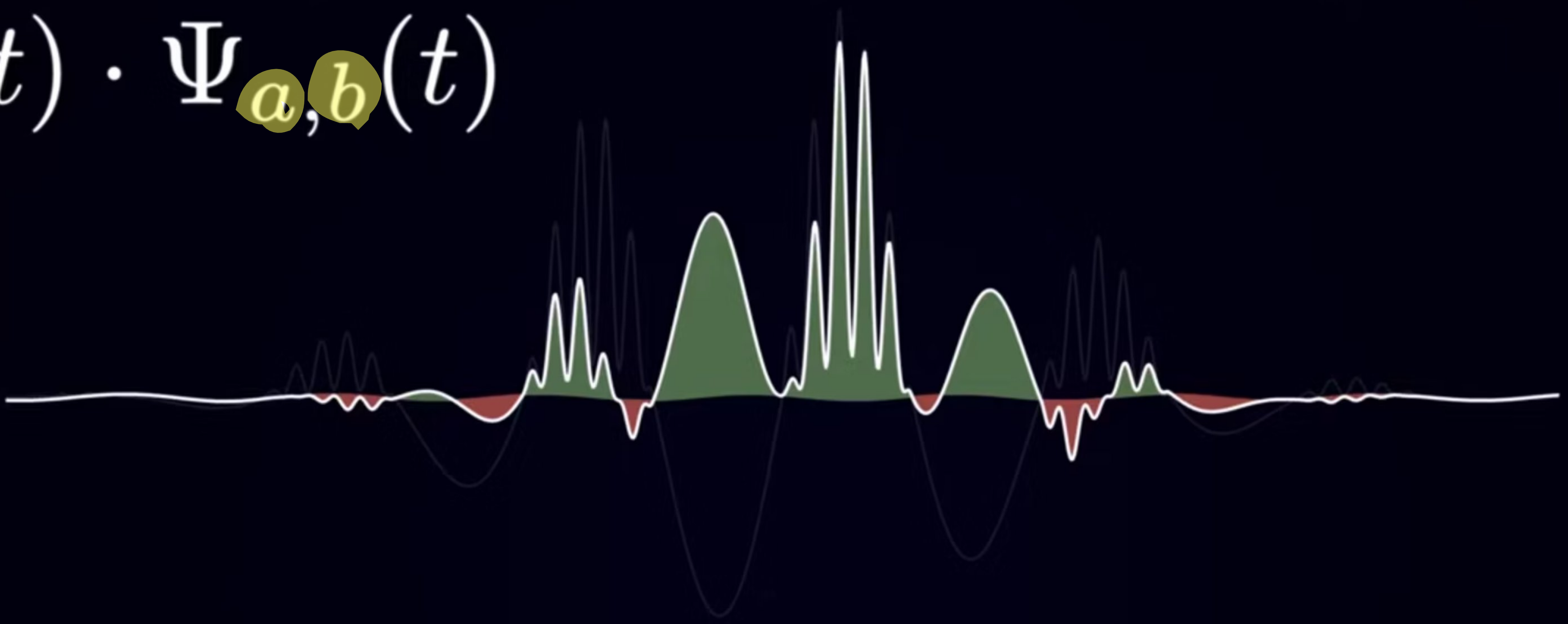


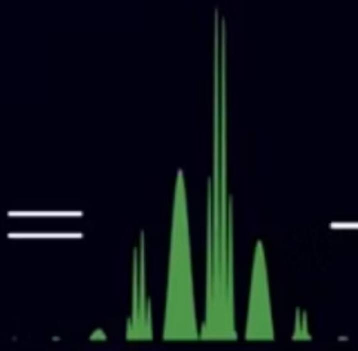


$$y(t) \cdot \Psi_{a,b}(t)$$



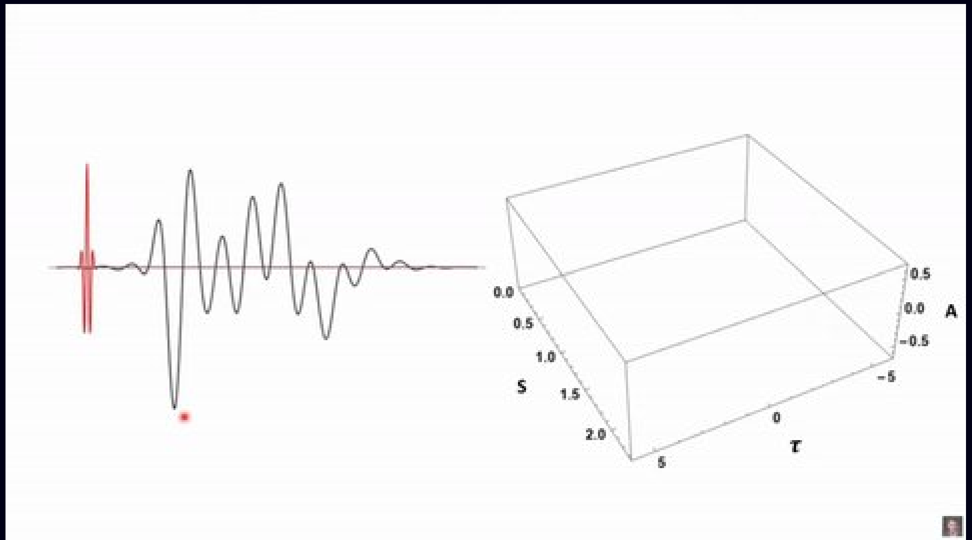
$$y(t) \cdot \Psi_{a,b}(t)$$



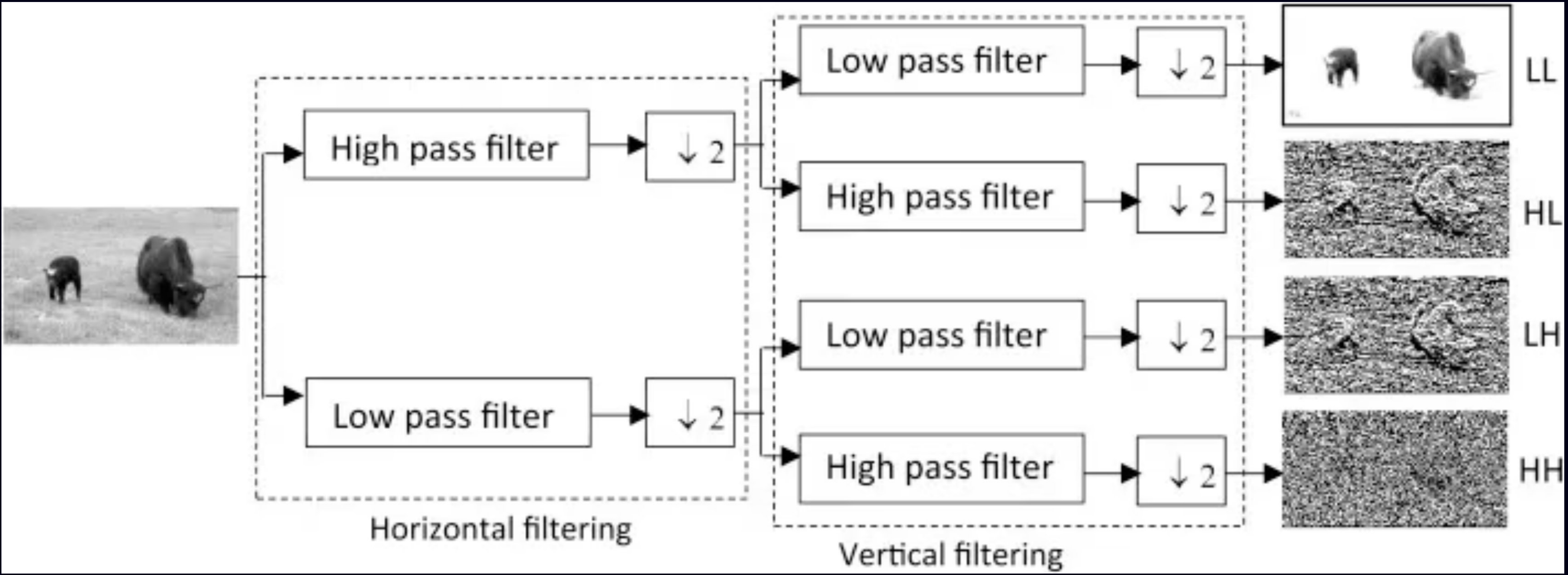
$$\int_{-\infty}^{+\infty} y(t) \cdot \Psi_{a,b}(t)dt =$$  $-$ 

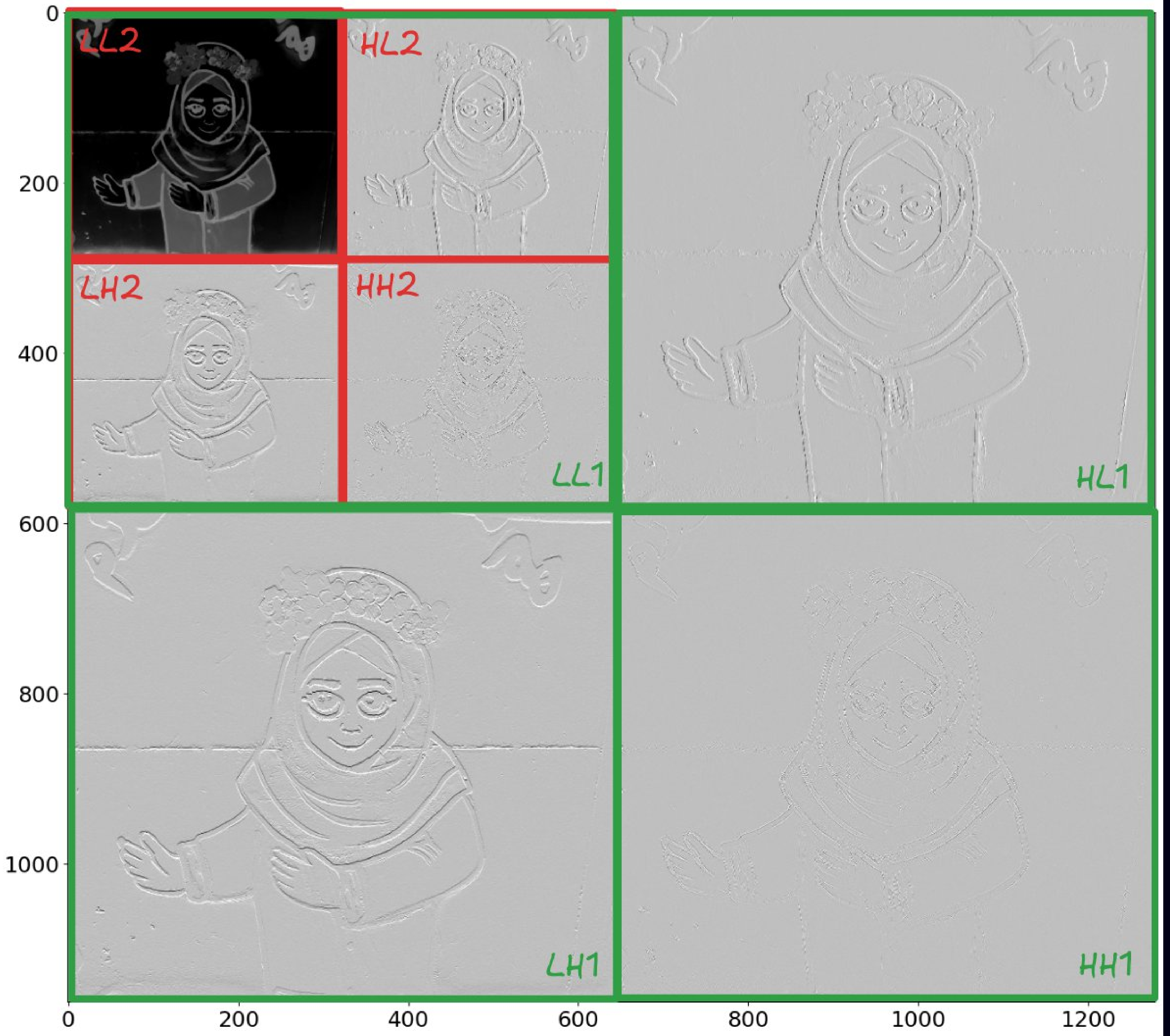
note : Think of a dot product in infinite dimension we see that a and b parameters that are the paremeters that change the moteher wavelete change so that we can capture all frequencies

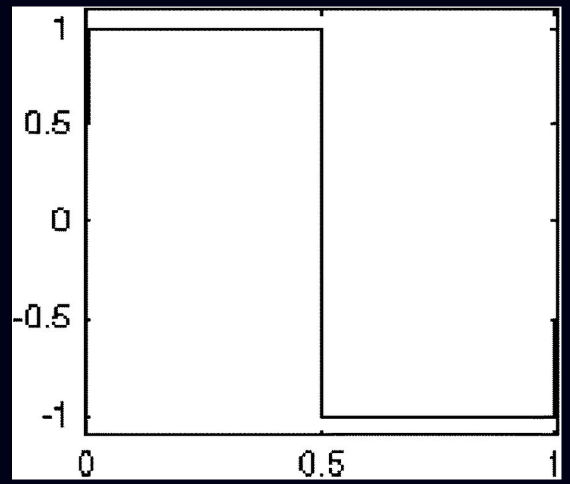
<Animation TIME />



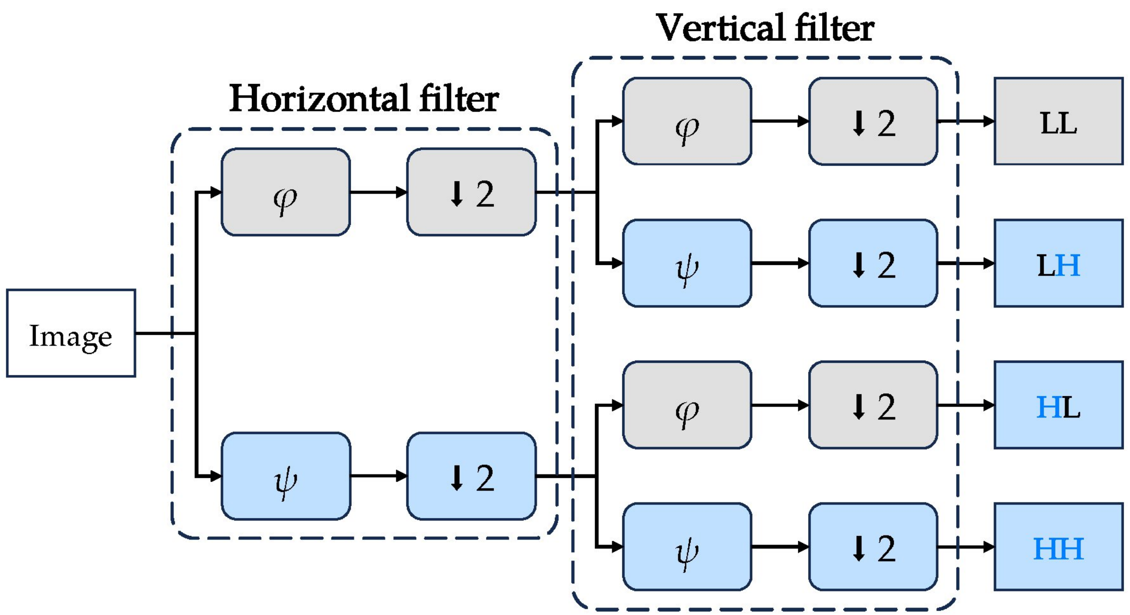
note : Think how Kernel in Constitutional neural network worked!
(e.g. character recognition)


High pass filter
↓ 2
Low pass filter
↓ 2
Horizontal filtering
Low pass filter
↓ 2
High pass filter
↓ 2
Low pass filter
↓ 2
High pass filter
↓ 2
Vertical filtering
LL
HL
LH
HH

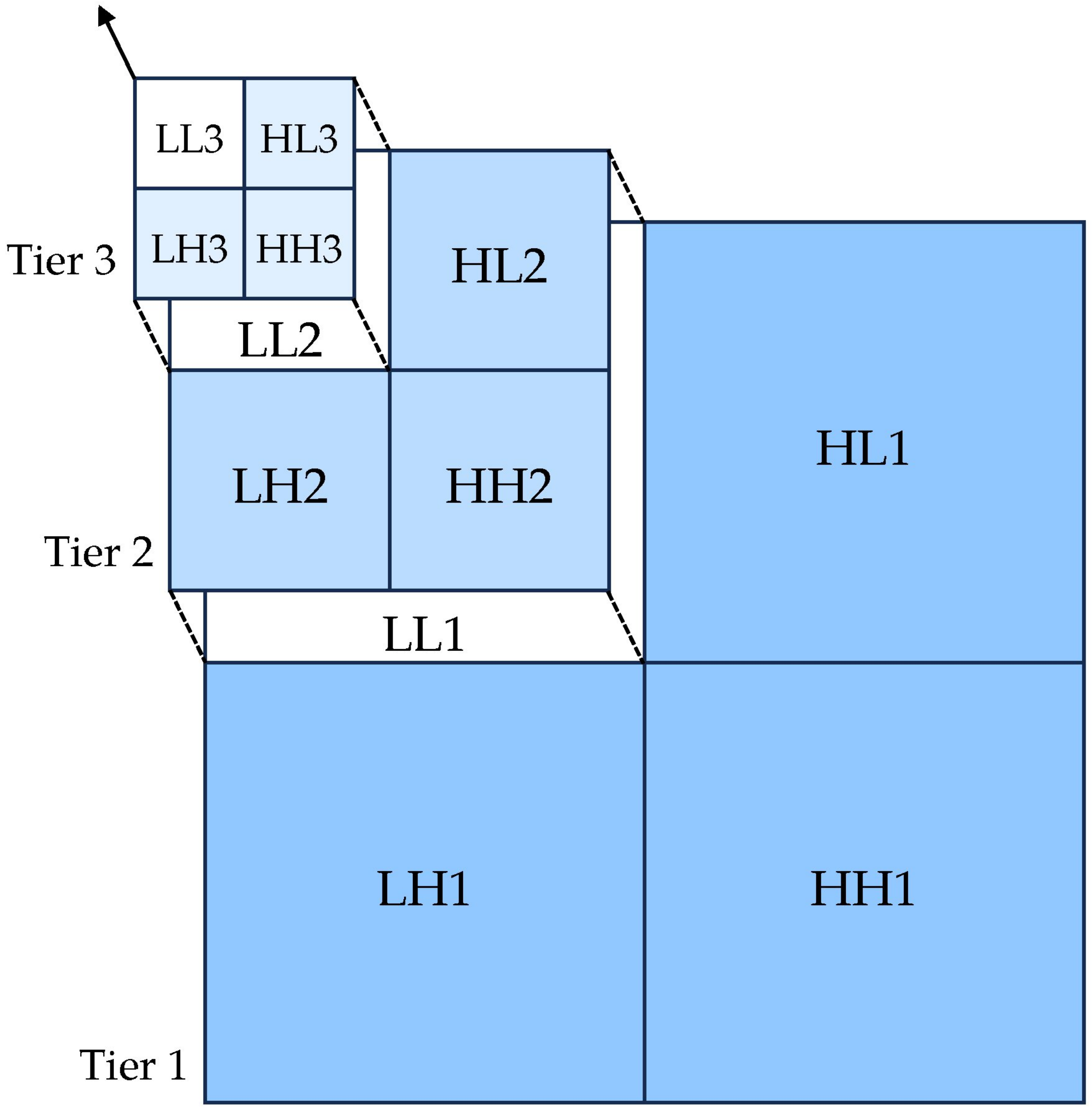




db1-in-the-Daubechies-family-wavelets-used



(**a**) Procedure of 2D-DWT.



(**b**) Hierarchy of 2D-DWT.

DWT Applications

Image Compression

Powers JPEG 2000 by removing unnecessary frequency bands moment-by-moment rather than entire frequency bands

De-noising

Reduces noise by attenuating coefficients below thresholds across all levels

Medical Imaging

Compresses and recovers medical image data without loss, enabling tele-monitoring

Signal Analysis

Examines time-frequency characteristics with variable resolution for complex signals

The DWT's O(N) time complexity (faster than FFT's O(N log N)) makes it ideal for real-time applications

<CODE TIME />

<Source TIME />

https://www.youtube.com/watch?v=jnxqHcObNK4&t=1652s

https://www.youtube.com/watch?v=jNC0jxb0Ox
E&list=PLMrJAkhIeNNT_Xh3Oy0Y4LTj0Oxo8GqsC