

Wavelet Transform for Image Processing

Zhang Jinrui¹

He Jiashun

Meng Jingyuan

Jiang Zishen

Mo Zian

August 5, 2025

¹alternative email: zhangjr1022@mails.jlu.edu.cn

Background and Motivation

- Image preprocessing is a crucial step in modern computer vision and machine learning.
- Models often require input images to have a fixed size and consistent format.
- Our goal is to convert any input image into a standardized grayscale format of size $2^N \times 2^N$.
- This not only simplifies downstream processing but also ensures compatibility with wavelet transform techniques, which rely on consistent input dimensions.

Wavelet Transform Overview

- The wavelet transform is a powerful mathematical tool for analyzing signals at multiple scales.
- Unlike the Fourier transform, which only reveals frequency content, wavelets provide both frequency and spatial localization.
- This is particularly useful in image processing where textures, edges, and fine structures need to be captured at different resolutions.
- Common wavelet families include Haar, Daubechies, and Coiflets — all benefit from inputs with dimensions that are powers of two.

Image Processing Workflow

- 1 **Image Loading and Grayscale Conversion:** The image is loaded using a Python imaging library and converted to grayscale. This reduces data complexity and focuses analysis on structural content.
- 2 **Rescaling with Aspect Ratio Preservation:** The image is resized using high-fidelity interpolation to ensure one side reaches the target length (2^N), without distortion.
- 3 **Centered Cropping:** The resized image is cropped to $2^N \times 2^N$, ensuring input uniformity without compromising important visual features.
- 4 **Matrix Output:** The final image is converted into a matrix format suitable for mathematical operations, storage, and machine learning integration.

Theoretical Justification

- Many wavelet algorithms are optimized for inputs whose dimensions are powers of two — enabling efficient recursive decomposition.
- Improper dimensions can introduce edge effects or truncation artifacts, degrading transform results.
- LANCZOS interpolation is used due to its ability to preserve edges and textures during resizing, which is critical for wavelet-based analysis.
- Converting to a 2D matrix makes the image directly usable for NumPy operations and compatible with machine learning frameworks like TensorFlow and PyTorch.

Application Scenarios

- **Educational Tools:** Demonstrate how images are converted, processed, and decomposed in the frequency domain.
- **Scientific Research:** Used in experiments where consistent image input is necessary, such as style transfer or anomaly detection.
- **Data Preprocessing:** Helps in preparing inputs for neural networks, especially for tasks that benefit from texture analysis like medical imaging or remote sensing.
- **Feature Engineering:** Enables extraction of localized frequency features for traditional machine learning models.

Extension Opportunities

- **Multi-channel Support:** Extend to RGB or multi-spectral image processing for richer feature representations.
- **Batch Automation:** Apply the transformation to entire datasets, allowing integration into large-scale pipelines.
- **Visualization Tools:** Incorporate graphical interfaces to preview the original vs. transformed image and to display wavelet decomposition layers.
- **Parameter Tuning:** Allow users to experiment with different interpolation methods or wavelet bases interactively.

Conclusion

- The image conversion program serves as a foundational tool that bridges raw image data and advanced analysis techniques.
- Its adherence to standardized formats makes it ideal for educational purposes, algorithm development, and production pipelines.
- Future development can make it even more powerful by supporting color analysis, batch operations, and intuitive visual feedback.

Standard Haar Decomposition

- sdlfkj
- sDF
- sdf

sdfg

- sdlfkj
- sDF
- sdf

sadsgkjh

- sdlfkj
- sDF
- sdf

sdf

- sdfkjk
- sDF
- sdf

Basic Idea

$$\sum_{k=1}^n \hat{l}_k^4 \leq \left(\sum_{k=1}^n \hat{l}_k^2 \right)^2 = \left(\sum_{k=1}^n l_k^2 \right)^2$$

$$\min_{\hat{l}_k \in l^2 \text{ st. } \sum_{k=1}^n \hat{l}_k^2 = \sum_{k=1}^n l_k^2} \left(- \sum_{k=1}^n \hat{l}_k^4 \right)$$

Basic Idea

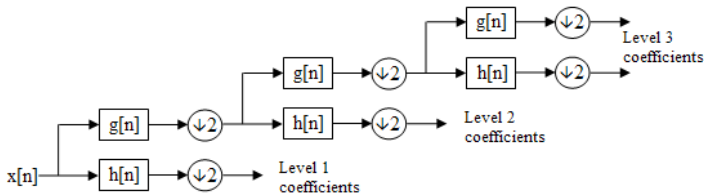


Figure: Cascading

$$\min_{\Phi} \left(- \sum_{k=1}^n \hat{l}_k^4 \right)$$

- Condense the energy to as less coefficients as possible.

Results

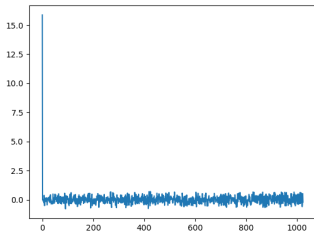


Figure: Haar 2x2 filter bank random input. Compression Rate 0.2568

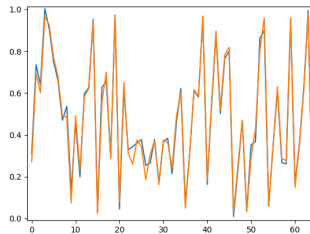


Figure: Haar 2x2 filter bank random input. Total average energy loss 0.0009

- random sequence would take more information entropy so have a low compress rate.

Results

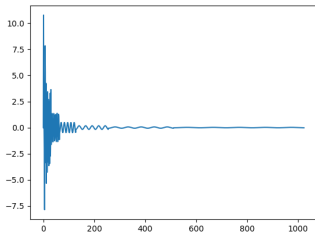


Figure: Haar 2x2 filter bank sin input frequency. Compression Rate 0.9287

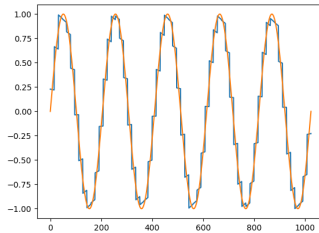


Figure: Haar 2x2 filter bank sin input reconstruction. Total average energy loss 0.0104

- a smooth signal not have much information entropy, so will have a high compress rate.

Results

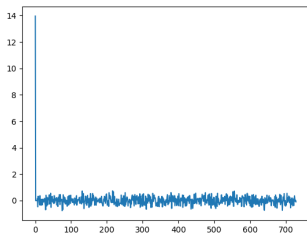


Figure: Haar 3x3 filter bank random input. Compression Rate 0.1906

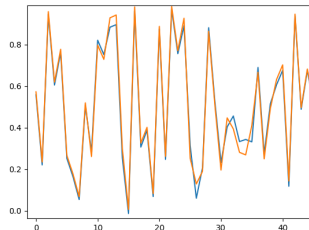


Figure: Haar 3x3 filter bank random input. Total average energy loss 0.0008

- same random sequence same reason.

Results

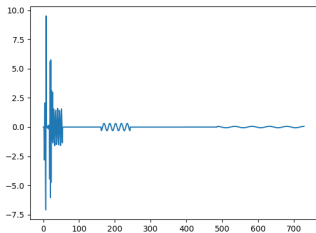


Figure: Haar 3x3 filter bank sin input frequency. Compression Rate 0.7750

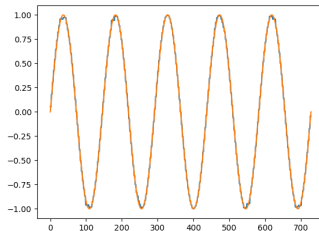


Figure: Haar 3x3 filter bank sin input reconstruction. Total average energy loss 0.0007

- same smooth signal same reason.

sadsgkjh

- sdIfkj
- sDF
- sdf

sdf

- sdlfkj
- sDF
- sdf