

USER MANUAL

Fourier-Domain High-Boost Image Filtering

Team Members:

Srinija Enimireddy
Atharva Wakulkar
Suhas H

Abstract

This manual provides comprehensive instructions for installing, configuring, and executing the Fourier High-Boost Filter application. The software allows users to sharpen images using frequency domain techniques, featuring a GUI for interactive parameter tuning and batch scripts for automated processing.

Contents

1	System Requirements	3
2	Installation Guide	3
2.1	Set Up the Environment	3
2.2	Install Dependencies	3
3	Project Directory Structure	4
4	How to Run: GUI Application	5
4.1	Launch	5
4.2	Workflow	5
5	Testing and Verification	5
5.1	Automated Unit Tests (Pytest)	5
5.2	Rigorous Core Module Testing (Batch Processing)	6
6	Troubleshooting	6

1 System Requirements

Before proceeding, ensure your system meets the following specifications:

- **Operating System:** Windows 10/11, macOS, or Linux (Ubuntu 20.04+ recommended).
- **Python Version:** Python 3.10 or higher.
- **Tested Environment:** Validated on **Python 3.13**.
- **Supported Image Formats:** JPG, JPEG, PNG, TIFF, GIF, BMP, AVIF.

2 Installation Guide

2.1 Set Up the Environment

It is highly recommended to use a virtual environment to avoid dependency conflicts. The instructions below are tailored for the tested **Python 3.13** environment.

Option A: Using Conda (Recommended)

```
# Create a new environment with Python 3.13
conda create -n fft_hb python=3.13 -y
conda activate fft_hb
```

Option B: Using venv (Standard Python)

```
# Windows
python -m venv venv
venv\Scripts\activate

# macOS / Linux
python3.13 -m venv venv
source venv/bin/activate
```

2.2 Install Dependencies

Navigate to the project root directory containing `requirements.txt` and run:

```
pip install -r requirements.txt
```

Core Dependencies installed:

- `numpy`: For FFT and matrix operations.
- `matplotlib`: For plotting spectra and saving diagrams.
- `Pillow`: For extensive image I/O handling (including AVIF/TIFF).
- `ttkbootstrap`: For the modern GUI theme.
- `pytest`: For the automated test suite.

3 Project Directory Structure

The project adheres to the following modular structure:

```
FOURIER_HIGHBOOST_FILTER_PROJECT/
    main.py                                # Entry point for the GUI application
    pytest.ini                             # Configuration for test suite
    requirements.txt                      # Python dependencies
    README.md                               # Project description and usage guide
    core/
        fft_engine.py                     # Wrappers for FFT/IFFT and shift operations
        filters.py                        # Generation of Gaussian/Butterworth masks
        highboost.py                      # High-boost formula logic
        color_processing.py              # RGB channel splitting and recombination
        __init__.py
    gui/
        interface.py                      # User Interface
        callbacks.py                      # Main Window layout (ttkbootstrap)
        utils.py                           # Event handling (Load, Run, Save)
        __init__.py
    io_utils/
        image_handler.py                 # GUI helper functions
        file_utils.py                    # Pillow-based reading/writing
        __init__.py
    visuals/
        plots.py                          # Timestamped filename generation
        __init__.py
    scripts/
        batch_demo.py                   # Visualization Tools
        demo_plots.py                  # Matplotlib utilities for spectra and masks
        debug_intermediates.py         # Debugging & Demo Tools
        debug_g_shifted_symmetry.py   # Diagnostic tool for pipeline steps
    tests/
        test_fft_engine.py
        test_filters.py
        test_highboost.py
        test_color_processing.py
        test_io_utils.py
        test_invariants.py
        test_visuals.py
        test_pipeline_large_image.py
        test_pipeline_basic.py
    data/                                  # Folder for sample images
    docs/                                  # Documentation files
    results/                                # Batch processing results (spectra, logs)
```

4 How to Run: GUI Application

The primary interface for this project is a modern, Tkinter-based GUI.

4.1 Launch

From the project root directory, execute:

```
python main.py
```

4.2 Workflow

1. **Open Image:** Click the blue “Open Image” button.
 - Navigate to the `data/` folder or any local directory.
 - Select an image (JPG, PNG, TIFF, BMP, GIF, AVIF).
 - The left pane will display the *Original Image*.
2. **Configure Parameters:**
 - **Filter Type:** Select *Gaussian* (smoothest), *Butterworth* (tunable), or *Radial* (sharp cutoff).
 - **Cutoff (D_0):** Input the radius of the low-pass mask (e.g., 20.0 to 50.0). Smaller values blur more of the base, affecting the boost context.
 - **Boost Factor (r):** Input a value > 1.0 .
 - $r = 1.0$: No boost (Original Image).
 - $r = 1.5 - 2.0$: Subtle sharpening.
 - $r > 3.0$: Aggressive edge enhancement.
 - **Order (n):** (Butterworth only) Controls the steepness of the filter roll-off.
3. **Execute:** Click the green “Run High-Boost” button.
 - The application computes the 2D FFT, applies the mask $H_b = r + (1 - r)L$, and performs the Inverse FFT.
 - The right pane displays the *High-Boosted Result*.
4. **Save:** Click the purple “Save Output” button to export the result to the `results/` folder (or a custom path).

5 Testing and Verification

The project employs a comprehensive testing strategy to ensure mathematical correctness and stability.

5.1 Automated Unit Tests (Pytest)

The project includes a robust test suite to verify the mathematical correctness of the FFT implementation and the logic of the high-boost filter.

Execution:

```
pytest -v
```

What is Verified?

- **Hermitian Symmetry:** Ensures that the filter masks in the frequency domain are symmetric so that the inverse FFT yields a purely real image (no complex artifacts).
- **Channel Independence:** Confirms that R, G, and B channels are processed separately without color bleeding.
- **Boundary Conditions:** Tests edge cases like $D_0 = 0$ (Pinhole logic).

5.2 Rigorous Core Module Testing (Batch Processing)

Important Note: The batch processing script is **not** a standard operational mode for general users. It was designed specifically to rigorously test the core mathematical modules ('core/') across a dataset of images to verify stability under varied conditions.

This module automates the entire pipeline and generates deep diagnostic artifacts (Frequency Spectra, Phase Maps, and Mask Heatmaps) to allow visual inspection of the intermediate mathematical steps.

Execution:

```
python -m scripts.batch_demo
```

Diagnostic Outputs: This script creates a timestamped folder inside `results/` containing:

- **Processed Images:** The final high-boosted outputs.
- **Spectra Plots:** `fft_spectrum.png` showing the log-magnitude frequency domain.
- **Mask Visuals:** `filter_mask.png` and `highboost_mask.png` (Heatmaps of the filters applied).
- **CSV Report:** `results.csv` summarizing parameters (D_0, r) and file paths for all processed images.

6 Troubleshooting

1. Error: ModuleNotFoundError or ImportError

- *Cause:* Dependencies are not installed or the virtual environment is not activated.
- *Fix:* Run `pip install -r requirements.txt` inside your active environment.

2. Error: _tkinter.TclError: no display name (Linux/WSL)

- *Cause:* Missing system-level Tkinter libraries.
- *Fix:* Run `sudo apt-get install python3-tk`.

3. Issue: Image Not Loading

- *Cause:* Unsupported format or corrupted file.
- *Fix:* Ensure the image is one of the supported formats (PNG, JPG, TIFF, BMP, GIF, AVIF). Try converting it to PNG if issues persist.

4. Issue: Output is Black or All White

- *Cause:* Extreme boost factor (r) or incorrect cutoff (D_0).
- *Fix:* Reset parameters to defaults ($D_0 = 50, r = 1.5$). Ensure $r > 1.0$.