

Phase 1: The Entry Point & Interface

Goal: Understand how the application starts and how the window is built.

1. The Starter Script (main.py)

This is the simplest file in the project. It is the "key" that starts the engine.

```
from gui.interface import launch_app

if __name__ == "__main__":
    launch_app()
```

Explanation:

- It imports the `launch_app` function from our GUI folder.
 - The `if __name__ == "__main__":` block ensures the app only runs when you specifically tell Python to run this file.
-

2. The User Interface (gui/interface.py)

This file builds the visual window. It uses a library called `ttkbootstrap` to make the window look modern (Dark Mode).

A. Setting up the Window

The class `HighBoostApp` represents our main window. In the initialization step (`__init__`), we define the "State Variables" the data the app needs to remember.

```
class HighBoostApp(ttk.Window):
    def __init__(self):
        # Set Window Title and Size
        self.title("Fourier High-Boost Filter")
        self.geometry("1200x700")

        # 1. Placeholders for Images
        self.image_np = None          # Will hold the Original Image
        self.output_np = None         # Will hold the Result Image

        # 2. User Inputs (Variables linked to Sliders/Boxes)
        self.filter_type = StringVar(value="gaussian")
        self.D0_val = DoubleVar(value=50.0) # Cutoff Frequency
        self.r_val = DoubleVar(value=1.8)   # Boost Factor
```

What this does: It prepares the "memory" of the application. It creates variables for D_0 and r that automatically update whenever the user types in the GUI boxes.

B. Building the Layout

The `_build_layout` function splits the window into two main sections:

1. **Left Panel (Controls):** This contains the buttons and inputs.

```
# Filter Type Dropdown
opt = ttk.OptionMenu(control, self.filter_type, "
    Gaussian", ...)

# Numeric Inputs
ttk.Entry(control, textvariable=self.D0_val)
ttk.Entry(control, textvariable=self.r_val)

# Buttons
ttk.Button(text="Open Image", command=
    open_image_callback)
ttk.Button(text="Run High-Boost", command=
    run_highboost_callback)
```

Note how we link the buttons to "callbacks". A callback is just a function that runs when the button is clicked.

2. **Right Panel (Display):** This area is divided into two side-by-side boxes.

- **Left Box:** Displays the *Original Image*.
- **Right Box:** Displays the *Processed Result*.

C. Displaying Images

We have a helper function `preview_original` that takes a raw NumPy array (the image data) and puts it on the screen.

```
def preview_original(self, arr):
    # 1. Save a copy of the raw data
    self._orig_np = arr.copy()

    # 2. Convert NumPy array to a Picture format (Pillow)
    pil = Image.fromarray(arr.astype(np.uint8))

    # 3. Draw it on the screen
    tking = ImageTk.PhotoImage(pil)
    self.orig_canvas.create_image(0, 0, image=tkimg)
```

Summary of Phase 1: We have successfully created a window, set up the input fields for our math parameters (D_0, r), and prepared the areas where images will be shown. No image processing has happened yet we have just built the control center.