Industrial Project 2024 Part 01

Comprehensive Guide for Project Execution and Usage

Group A:

Sonain Jamil, Kasem Amnuayrotchanachinda, Muhammad Turab

sonainjamil@ieee.org

November 20, 2024

Contents

1 Introduction

The **Industrial Project 2024 Part 01** is designed for processing and analyzing hyperspectral and multispectral image data. Utilizing Python, it implements interactive visualization, data binning, preprocessing, and spectral transformations to enhance understanding of spectral properties and derivatives. This project is well-suited for research and industrial applications involving spectral data and image analysis.

2 Project Features

- **Spectral Image Preprocessing:** Converts raw spectral data to reflectance images and allows for cropping and binning.
- **Data Binning:** Includes spatial and spectral binning to reduce data dimensions while preserving essential spectral features.
- **Interactive Visualization:** Provides an RGB visualization and an interactive spectral plot for a selected region, with options to view normalized spectra and its first and second derivatives.
- **Configurable Pipeline:** Leverages a `config.yaml` file for flexible input paths, parameters, and settings, enabling easy customization for different datasets.

3 Installation and Setup

3.1 Requirements

- Python 3.8 or higher.
- Required Python libraries:
  - `numpy`
  - `matplotlib`
  - `PyYAML`
  - `opencv-python`
  - `scikit-learn`

3.2 Setup Instructions

1. Clone the repository to your local machine:

```
git clone https://github.com/sonainjameel/Industrial_Project_2024.
    git
cd Industrial_Project_2024
```

2. Install the required dependencies:

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

3. Ensure you have the correct input data and configurations as specified in the `config.yaml` file.

4 Configuration

The project relies on a `config.yaml` file for setting input paths and processing parameters. Below is an example configuration:

```yaml
paths:
  header_file: "/path/to/header/file.hdr"
  dark_reference: "/path/to/dark_reference.hdr"
  white_reference: "/path/to/white_reference.hdr"
  output_dir: "/path/to/output/directory"
  spectral_binned_file: "/path/to/spectral_binned_image.hdr"
parameters:
  spatial_bin_size: 2
  spectral_bin_size: 2
  target_wavelengths: [1496.9, 1301.1, 1104.8]
  reflectance_factor: "1" # for reference image; "0.448 * 0.95" for charred
      wood samples
  crop_params:
    x1: 20
    y1: 950
    x2: 300
    y2: 1220
```

Listing 1: Example config.yaml File

5 Usage

To run the main script and execute the pipeline:

```
python3 main.py config.yaml
```

Listing 2: Run the Main Script

This command will process the specified hyperspectral image, perform data binning, and launch an interactive visualization window.

6 Project Structure

```
Industrial_Project_2024/
|-- config.yaml                        # Configuration file for paths and parameters
|-- main.py                            # Main script to run the pipeline
|__ my_project/                       # Core project folder
    |-- __init__.py                    # Initialize package
    |-- io_utils.py                    # I/O functions for loading and saving data
    |-- preprocessing.py               # Functions for preprocessing images
    |-- binning.py                     # Functions for spatial and spectral binning
    |__ interactive_visualization.py # Interactive visualization functions
```

7 Key Modules

- `io_utils.py`: Handles loading and saving of spectral image data.
- `preprocessing.py`: Contains image preprocessing functions, including normalization and reflectance conversion.
- `binning.py`: Implements spatial and spectral binning to reduce data dimensions.
- `interactive_visualization.py`: Provides an interactive visualization interface, allowing users to view an RGB image and explore spectral data across selected regions.

8 Interactive Visualization

The interactive visualization window allows users to explore:

- **Mean Spectrum:** Displays average reflectance.
- **First Derivative:** Highlights changes across wavelengths.
- **Second Derivative:** Indicates areas of rapid spectral change.

9 Testing and Results

The following test cases were conducted:

- Processed images with varying resolutions to ensure the binning algorithms work correctly.
- Verified configuration file flexibility with multiple datasets.

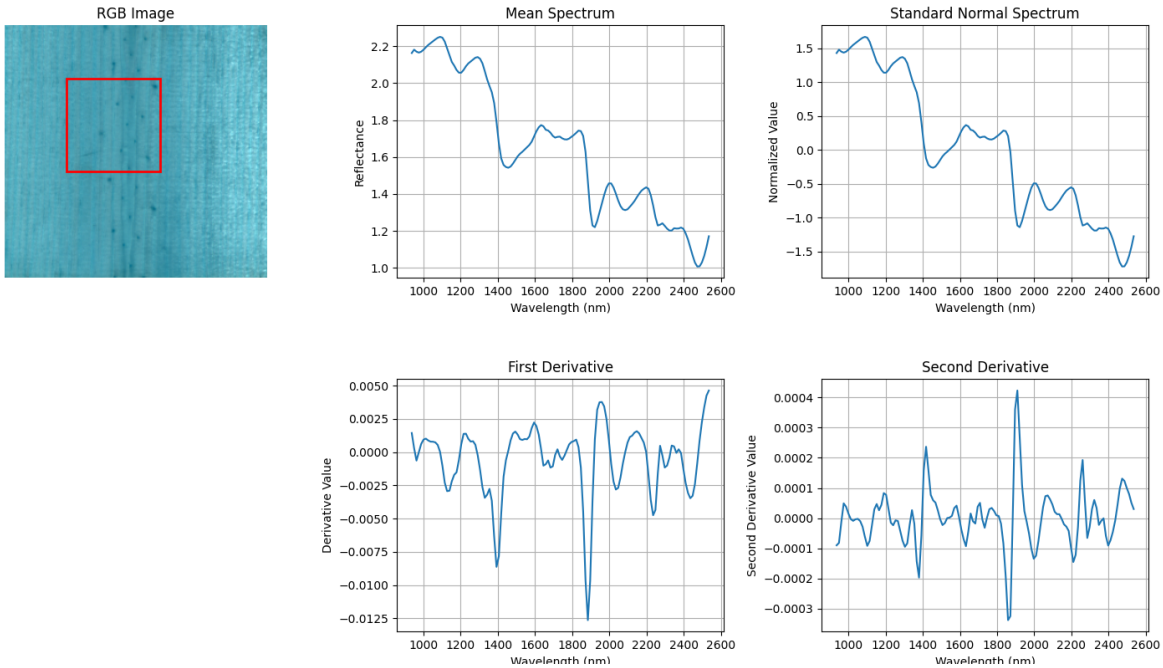

Figure 1: Example visualization of the output.

10 References

- Python Documentation: https://www.python.org/doc/
- Libraries: NumPy, OpenCV, Matplotlib, PyYAML, Scikit-Learn.