Advanced Spectral Imaging: Homework 2

Yuya Haga

December 23, 2024

1 Tasks #1. Colab/Python and spectral files.

1.1 Specim IQ

1.1.1 Preview

A spectral image by SpecimIQ was loaded by the Python script shown in Code 4 and converted to RGB preview by Code 5. The gray scale preview and RGB preview are shown in Figure 1.

Gray preview of the image was created by the following code:

```python
from pathlib import Path

path = Path(’ASI course 2024/group3/Session1/Specim scanner/Color_checker_8_binning/capture/solutions_scan_0110’)
spectral_image, envi_header = load_spectral_image(path)
plt.imshow(spectral_image[:, :, 100], cmap="gray")
```

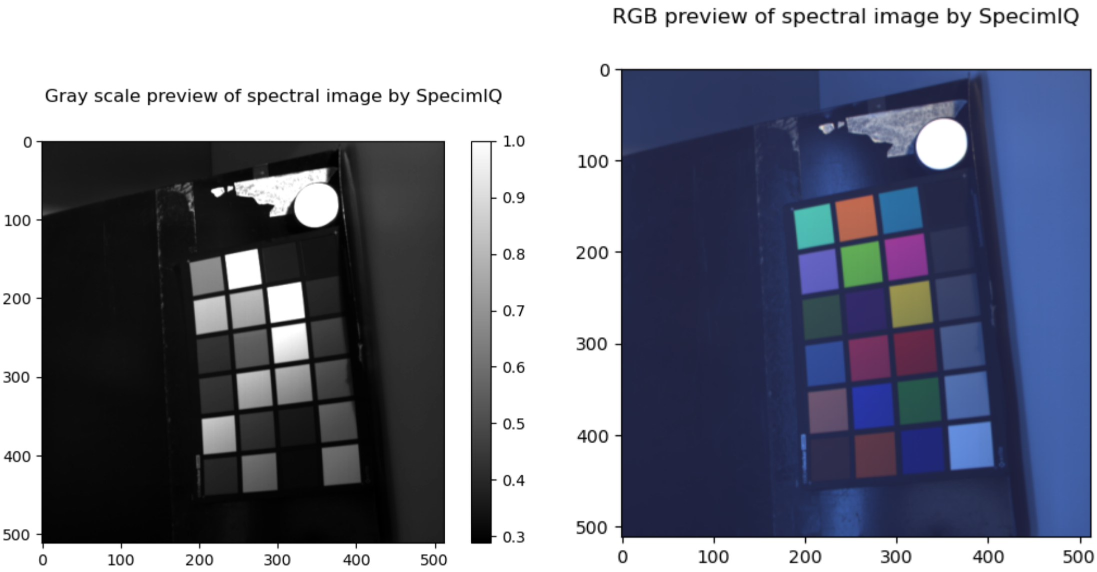
Code 1: Show gray scale preview

RGB preview of the image was created by the following code:

```python
rgb_view = reconstruct_rgb_envi(spectral_image, envi_header)
plt.imshow(rgb_view)
```

Code 2: Show RGB preview

Functions used in Code 1 and Code 2 are defined in Code 4 and Code 5.


Gray scale preview of spectral image by SpecimIQ
0
100
200
300
400
500
0
100
200
300
400
500
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
RGB preview of spectral image by SpecimIQ
0
100
200
300
400
500
0
100
200
300
400
500

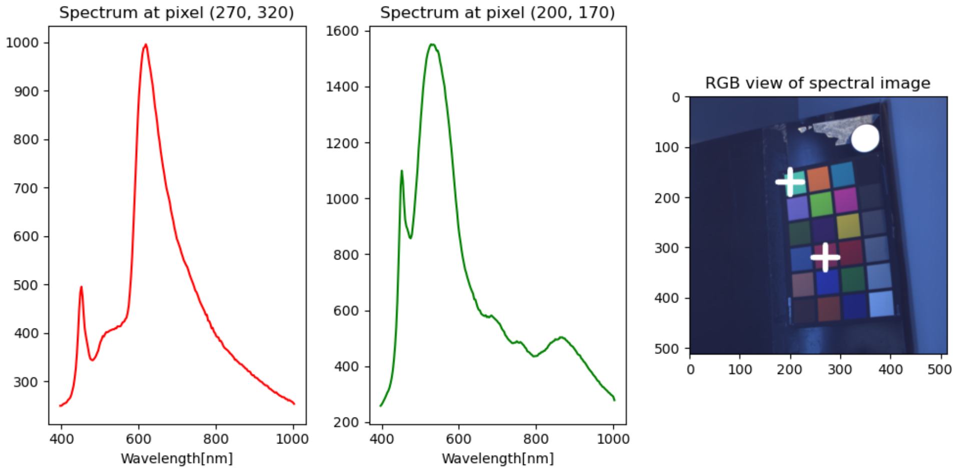
(a) Gray scale preview

(b) RGB preview

Figure 1: Previews of Specim IQ

1.1.2 Plot pair of spectra in one

Figure 2: Plot pair of spectra in one

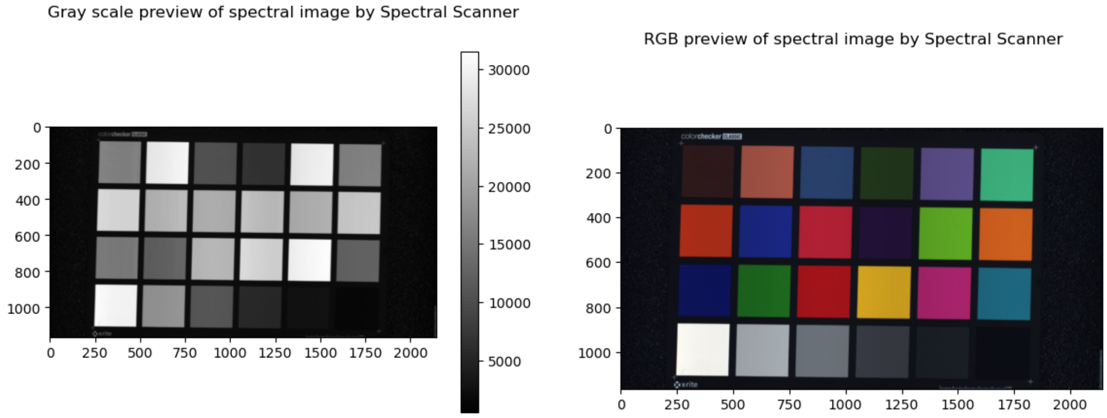


1.2 Spectral Scanner

Spectral image by Spectral Scanner was also processed by the same Python script. The gray scale preview and RGB preview are shown in Figure 3.

The same codes, with diffrent path in Code 1 and Code 2 were used to create the previews.

1.2.1 Preview



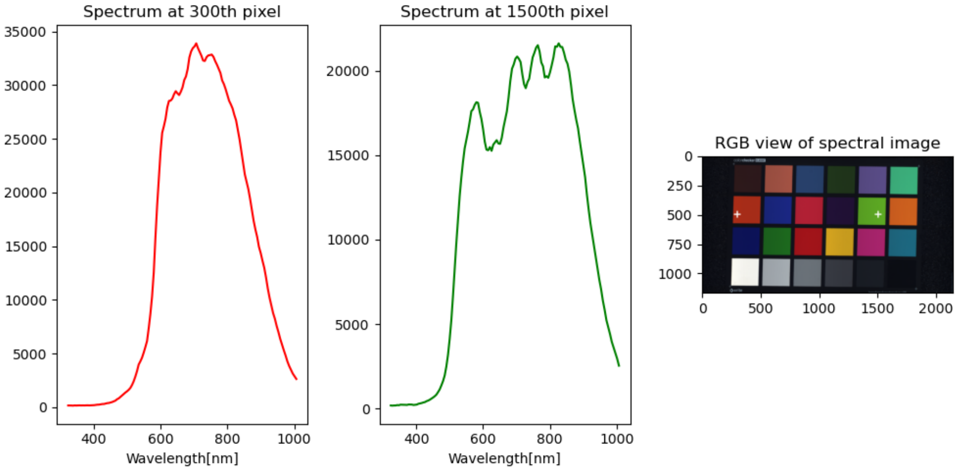
(a) Gray scale preview

(b) RGB preview

Figure 3: Previews of Spectral Scanner

1.2.2 Plot pair of spectra in one

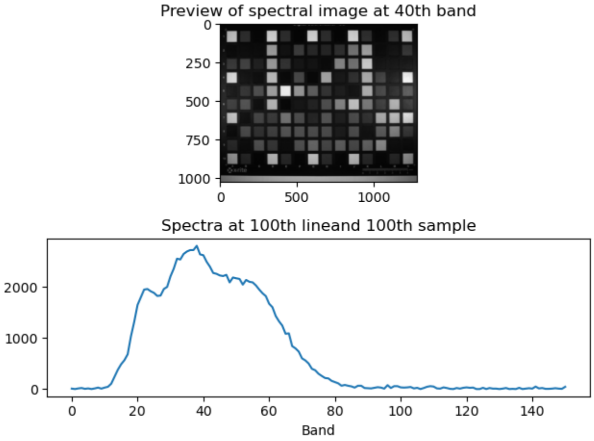
Figure 4: Plot pair of spectra in one



2 Tasks #2. Open ENVI from Japanese spectral camera

A spectral image by Japanese camera was loaded by the Python script shown in Code 6. The gray scale preview and RGB preview are shown in Figure 5.

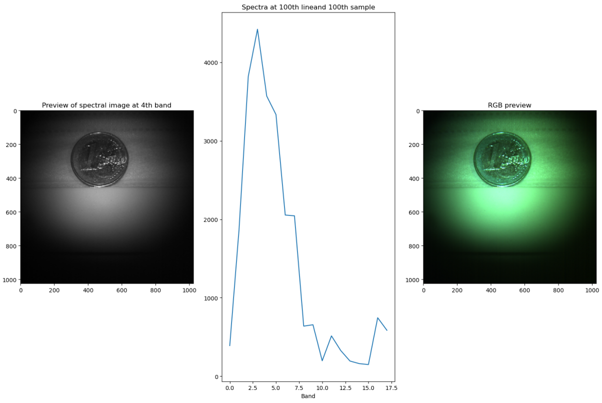
Figure 5: Gray scale preview of Japanese camera



3 Tasks #3. Open ENVI from other byte order

A spectral image of coin was loaded by the Python script shown in Code 7. The gray scale preview and RGB preview are shown in Figure 6. Function `load_envi_header` is defined in Code 4.

Figure 6: Previews of coin



4 Tasks #4. Save ENVI spectral image with BIP

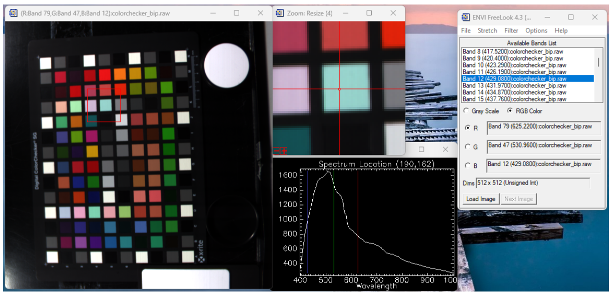
Original image was loaded from the ENVI format with the interleave format BIL. The image was saved with the interleave format BIP. The Python script is shown in Code 8.

Function `load_spectral_image` in 8 is defined in Code 4.

The header file of the BIP image is shown in Code 3. There are two changes from original one: description and interleave method.

The preview by FreeLook is shown in Figure 7.

Figure 7: Preview of BIP image by FreeLook



Band 8 (417.5200):colorchecker_bip.raw
Band 9 (420.4000):colorchecker_bip.raw
Band 10 (423.2900):colorchecker_bip.raw
Band 11 (426.1900):colorchecker_bip.raw
Band 12 (429.0800):colorchecker_bip.raw
Band 13 (431.9700):colorchecker_bip.raw
Band 14 (434.8700):colorchecker_bip.raw
Band 15 (437.7600):colorchecker_bip.raw

○ Gray Scale ● RGB Color

● R Band 79 (625.2200):colorchecker_bip.raw
○ G Band 47 (530.9600):colorchecker_bip.raw
○ B Band 12 (429.0800):colorchecker_bip.raw

Dims 512 x 512 (Unsigned Int)

```
ENVI
description = {Data transformed to BIP}
samples = 512
lines = 512
bands = 204
header offset = 0
file type = ENVI
data type = 12
interleave = BIP
sensor type = SPECIM IQ
byte order = 0
default bands = {70,53,19}
latitude = 0.00000000
longitude = 0.00000000
acquisition date = 29-09-2020
errors = none
binning = {1,1}
tint = 121
fps = 8.26446
wavelength = {
    397.32,
    400.20,
    403.09,
    405.97,
    408.85,
    ...
}
```

Code 3: Header file of BIP

5 Codes

The following Python scripts were used to complete the tasks. All of those codes are available in the GitHub repository [1].

```
from pathlib import Path

import numpy as np


def parse_envi_header(lines: list) -> dict[str, str]:
    """
    Parses ENVI file content into a structured dictionary
```

[1] https://github.com/8gaU8/ASI-Homeworks

```
    This code was written with Github Copilot
    """
    envi_data = {}
    in_block_key = None

    for line_org in lines:
        line = line_org.strip()

        # Skip empty lines
        if not line:
            continue

        # Handle multiline blocks
        if in_block_key:
            if line.endswith("}"):
                # Closing multiline
                envi_data[in_block_key] += line[:-1].strip()
                in_block_key = None
            else:
                # Continue multiline
                envi_data[in_block_key] += line
            continue

        # Key-value pair parsing
        if "=" in line:
            key, value = map(str.strip, line.split("=", 1))
            key = key.lower().replace(" ", "_")  # Normalize key format

            # Handle block values
            if value.startswith("{"):
                # Handle multiline block
                in_block_key = key
                # Remove opening ’{’
                value = value[1:].strip()
                if value.endswith("}"):
                    # Single-line block
                    envi_data[key] = value[:-1]
                    in_block_key = None
                else:
                    # Start multiline block
                    envi_data[key] = value
            else:
                # Single-line value
                envi_data[key] = value
    return envi_data


def load_envi_header(hdr_file: Path) -> dict[str, str]:
    """Loads ENVI header file."""
    with hdr_file.open(encoding="utf-8") as f:
        header_content = f.readlines()
    envi_header = parse_envi_header(header_content)
    return envi_header


def load_spectral_image(file_stem: Path) -> tuple[np.ndarray, dict[str, str]]:
    """Loads spectral image from ENVI format."""
    # Load ENVI header
```

```python
    hdr_file = file_stem.with_suffix(".hdr")
    envi_header = load_envi_header(hdr_file)

    # Load parameters
    interleave = str(envi_header["interleave"])
    lines = int(envi_header["lines"])
    samples = int(envi_header["samples"])
    bands = int(envi_header["bands"])
    data_type = int(envi_header.get("data type", 12))

    # Map ENVI data type to NumPy dtype
    data_type_map = {
        1: np.uint8,
        2: np.int16,
        3: np.int32,
        4: np.float32,
        5: np.float64,
        6: np.complex64,
        9: np.complex128,
        12: np.uint16,
        13: np.uint32,
        14: np.int64,
        15: np.uint64,
    }

    if data_type not in data_type_map:
        msg = f"Unsupported data type: {data_type}"
        raise ValueError(msg)

    dtype = data_type_map[data_type]

    # Load raw data
    raw_file = file_stem.with_suffix(".raw")
    with open(raw_file, "rb") as f:
        raw = np.fromfile(f, dtype=dtype)

    # define shape and transpose order by interleave method
    if interleave.upper() == "BIL":
        new_shape = (lines, bands, samples)
        axis_order = (0, 2, 1)
    elif interleave.upper() == "BIP":
        new_shape = (lines, samples, bands)
        axis_order = (0, 1, 2)
    elif interleave.upper() == "BSQ":
        new_shape = (bands, samples, lines)
        axis_order = (0, 2, 1)
    else:
        msg = f"Interleave {interleave} not supported."
        raise ValueError(msg)

    spectral_image = raw.reshape(new_shape)
    # change axis order to ’lines, samples, bands’
    spectral_image = np.transpose(spectral_image, axis_order)

    return spectral_image, envi_header
```

Code 4: Load ENVI format images

```python
import numpy as np


def search_closest_index(wavelengths: list[float], target_wavelength: float) -> int:
    """Finds the index of the closest wavelength to the target wavelength from value of envi header."""
    closest_index = int(np.argmin(np.abs(np.array(wavelengths) - target_wavelength)))
    return closest_index


def reconstruct_rgb_envi(
    spectral_image: np.ndarray,
    envi_header: dict[str, str],
    rgb_wavelengths: tuple[float, float, float] = (632.15, 528.03, 443.56),
):
    # """Reconstructs RGB image from spectral image with given three wavelengths."""
    # # Get wavelengths from header
    wavelengths = [float(data) for data in envi_header["wavelength"].split(",")]
    rgb_indeces = [search_closest_index(wavelengths, wl) for wl in rgb_wavelengths]
    lines, samples, _bands = spectral_image.shape
    rgb_view = np.empty((lines, samples, 3))
    # Reconstruct RGB image
    for idx, ch in enumerate(rgb_indeces):
        rgb_view[:, :, idx] = spectral_image[:, :, ch] / np.amax(
            spectral_image[:, :, ch]
        )
    return rgb_view
```

Code 5: Make RGB preview from given specral image

```python
from pathlib import Path

import matplotlib.pyplot as plt
import numpy as np

path = Path(
    "Camera from Japan/colorChecker.nh7"
)

# uint16 float32 #count=spatial_pixels*sample_lines*spectral_bands ’>u2’ numpy.uint16
with path.open("rb") as fopen:
    raw_image = np.fromfile(fopen, dtype=np.uint16)# Calculate number of bands from size of image
samples = 1280
lines = 1024

size_of_image = raw_image.shape[0]
bands = size_of_image // (samples * lines)
print(f"Number of bands: {bands}")

# Reshape the image to 3D array by bands, lines, samples
spectral_image = np.reshape(raw_image, (lines, bands, samples))
spectral_image = np.transpose(spectral_image, (0, 2, 1))
```

```

fig, axes = plt.subplots(2, 1, tight_layout=True)
axes[0].imshow(spectral_image[:, :, 40], cmap="gray")
axes[0].set_title("Preview of spectral image at 40th band")

axes[1].plot(spectral_image[100, 100, :])
axes[1].set_title("Spectra at 100th lineand 100th sample")
axes[1].set_xlabel("Band")
plt.show()
```

Code 6: Load Japanese camera

```
from pathlib import Path

import matplotlib.pyplot as plt
import numpy as np

root = Path("Image of coin (Senop camera)")
dat_path = root / "HSI_snapshot__20212101144836.dat"
header_path = root / "HSI_snapshot__20212101144836.hdr"

header = load_envi_header(header_path)

interleave = header["interleave"]
samples = header["samples"]
lines = header["lines"]
bands = header["bands"]
print(interleave)

with dat_path.open("rb") as fopen:
    raw_array = np.fromfile(fopen, dtype=">u2")

# Convert the string values of samples, lines, and bands to integers
samples = int(samples)
lines = int(lines)
bands = int(bands)

# Reshape the raw_array based on BSQ interleave
raw_image = raw_array.reshape((bands, samples, lines))
raw_image = np.transpose(raw_image, (1, 2, 0))


fig, axes = plt.subplots(1, 3, tight_layout=True, figsize=(15, 10))
axes[0].imshow(raw_image[:, :, 4], cmap="gray")
axes[0].set_title("Preview of spectral image at 4th band")

axes[1].plot(raw_image[100, 100, :])
axes[1].set_title("Spectra at 100th lineand 100th sample")
axes[1].set_xlabel("Band")

# show RGB preview
rgb_preview = raw_image[:, :, [1, 4, 6]].astype(np.float32)
rgb_preview /= np.max(rgb_preview, axis=(0, 1))
rgb_preview *= 2
rgb_preview = np.clip(rgb_preview, 0, 1)
axes[2].imshow(rgb_preview)
axes[2].set_title("RGB preview")

```

```python
plt.show()
```

Code 7: Load spectral image of coin

```python
from pathlib import Path

# load BIL spectral image
root = Path("Lectures+Exercises/LectureExercise #5, Freelook and bmp/
    Colorchecker 121 ms (for Freelook demo)/capture/")
si_path = root / "Colorchecker"
bil_spectral_image, bil_envi_header = load_spectral_image(si_path)

# reshape BIL to BIP
bip_spectral_image = bil_spectral_image.transpose(0, 1, 2)
# save BIP spectral image
bip_spectral_image.flatten().tofile("colorchecker_bip.raw")
```

Code 8: Load spectral image of coin