Journal of Imaging Science and Technology® 67(5): 1–6, 2023.

Efficient Hyperspectral Data Processing using File Fragmentation

C. Caruncho, P. J. Pardo, and H. Cwierz
Universidad de Extremadura, Merida, Badajoz, Spain
E-mail: ccaruncho@unex.es

Abstract. *In this article, we present a method for processing hyperspectral data in an easy and quick manner. We explain how we split the hyperspectral cube in different sections to be processed using fewer resources. We describe the processing, which includes extraction of the raw data along with white and black calibration data, calibration of the data and application of desired light source, color space, and gamma transformation. We then present a built-in software, including an easy interactive Graphical User Interface (GUI) that will allow fellow researchers to process hyperspectral images in a simple fashion.* © *2023 Society for Imaging Science and Technology.*
[DOI: 10.2352/J.ImagingSci.Technol.2023.67.5.050403]

1. INTRODUCTION

Hyperspectral imaging is a technique that involves capturing images of an object across a wide range of wavelengths, that can range from infra-red to ultraviolet regions of the electromagnetic spectrum. The result of this multiple captures is called "hyperspectral cube", a three-dimensional array of spectral data that describes the whole hyperspectral image [1]. The first two dimensions—width and height—correspond to the 2D images captured at different wavelengths, whereas the third dimension-depth-accounts for the wavelength itself. Each pixel in a hyperspectral image contains a spectrum of reflectance, that we can integrate along the spectrum to be set at a single value (Figure 1).

Although hyperspectral images are widely used in satellite photograph; our focus is in near images [2]. There is a key difference regarding the image processing of the two; satellite hyperspectral imaging uses raw data covering big areas, whereas near-hyperspectral imaging needs careful calibration for small areas. The results presented in this paper concern objects photographed with a hyperspectral camera within a 50 cm distance of the camera. The market offers different technologies, producing different qualities of data, however the developed software tool is hardware independent: it does not modify the quality but it can process all camera outputs.

The files storing the hyperspectral cube can usually be processed using the camera's built-in software. However, advanced users might want to explore the hyperspectral data to fit their specific needs and preferences. On the other hand, hyperspectral cameras are widely used by various professionals, primarily software engineers and technicians, as well as doctors, nurses, biologists, archaeologists, and others. Not all of these professionals have a strong computational background, nor do they necessarily have access to powerful equipment. We aim to offer an easy way to process these images on a personal computer and have quick on-hand results.

It is to be noted that this kind of imaging is extremely accurate, and therefore generates a large amount of information, which traduces on heavy files and process time consumption. It is precisely this characteristic that made us pursue a more efficient process using file fragmentation.

In this paper, we describe the image processing needed to obtain a hyperspectral image from the hyperspectral cube, that is, from the files furnished by the hyperspectral camera software. We will then present our software tool to do all this processing using a Graphical User Interface (GUI) that makes image handling easier [3].

2. HYPERSPECTRAL FILES

Using a hyperspectral camera to capture an object's image across the entire spectrum produces an Environment for Visualizing Images (ENVI) file format, consisting of a flat-binary raster file paired with an ASCII header file. The camera needs to be calibrated on white and black reflectances, which produces two more pairs of files. In our method, we expect to have them all (raw data and white/dark references) stored in the same folder. They are the files that will be processed and synthesised in one hyperspectral image.

Our approach consists in obtaining the hyperspectral image under a light source of our choice. For this, we need a light source file containing the light power spectral distribution. We recommend CIE illuminants in MatLab format (*mat).

Along with the illuminants, the CIE also proposes the response of the standard observer. We use the 2° observer to normalize the light source and the color tristimuli. In order to simulate the standard observer, a dedicated file (4 columns: one for wavelengths and three others for X, Y, Z color matching functions) is required [4]. It is noted that different observers can be used; CIE 10° observer would be the first variation but it can also be interesting to use this tool for evaluating new observer reflectances, vision defaults or enhancements.

Received May 12, 2023; accepted for publication Aug. 7, 2023; published online Sept. 13, 2023. Associate Editor: Peter Nussbaum.
1062-3701/2023/67(5)/050403/6/$25.00

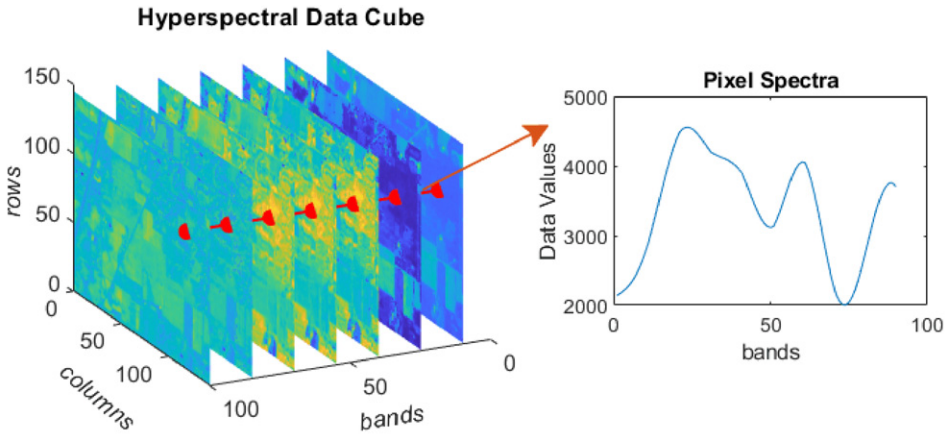

Figure 1. Hypercube and pixel spectrum example [3].

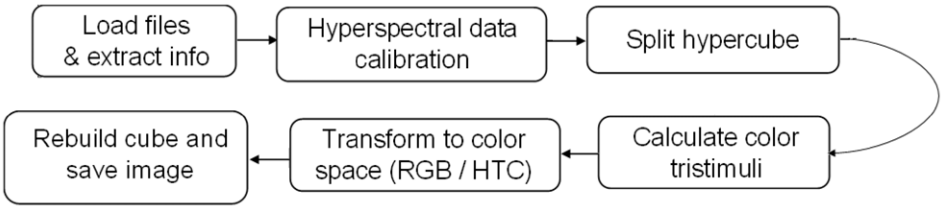

Figure 2. Scheme of hyperspectral data processing.

Calibration of images is crucial for a good image processing. Thus, the original data of the white reference target is also loaded as a MatLab file for adjusting the white calibration, since the reflectance of the target device can be diminished over time. It will be used to normalize the dark and white and references that were used for the photograph.

3. IMAGE PROCESSING

This present method for image processing has different stages (see Figure 2), which are detailed hereafter.

Once the working parameters are established, the hyperspectral camera files and CIE files (for the light source and standard 2° observer) will be loaded. Subsequently, the raw data will undergo calibration using white and dark references, and the cube will be partitioned into sections to facilitate computing and expedite execution.

When data is ready and formatted for optimising computing, the light source and the data tristimuli will be calculated. The resulting data will undergo a transformation from XYZ to RGB, along with a gamma transformation, depending on the device. This processing will generate a hyperspectral image that comprehensively captures all wavelengths and provides a precise view of the object. The cube will be separated into individual files, one for each frame (wavelength), enabling the user to gain a more comprehensive understanding of the photographed object.

3.1 *Settings*

Processing will be determined by the choice of the following parameters:

- number of parts for hypercube division; it is recommended to divide the cube into $h/500$ chunks, h being the horizontal resolution of the hyperspectral frame; but is up to the user to choose automatic ($h/500$) or splitting into 1–5 or 8 pieces;
- wavelengths in the range of interest; default is visible light (380–780 nm), but the user can go up to 1000 nm and thus engage infra-red wavelengths;
- step taken on the wavelength range, which will help lighten computation with small lost of quality;
- color space (device), that will determine the matrix transformation from XYZ to RGB. Two options are given: sRGB for standard PC monitor and htc for HTC-Vive headset. More devices and color spaces will be included soon.

3.2 *Load Files and Set Up Image Processing*

Image processing needs the following files:

- ENVI (flat-binary raster) files with corresponding ASCII header files; these are the files issued by the hyperspectral camera. For correct processing, two calibration files (that we call dark and white reference) are needed along with the raw data
- standard observer (2° or 10° CIE-recommended) file containing X, Y, Z reflectances for each wavelength [4];
- light source file (CIE-recommended) file containing the illuminant power spectral density. This file will allow the user to switch lights, changing the illuminant used for the photograph;
- white calibration reference file (fuzzy reference pattern).

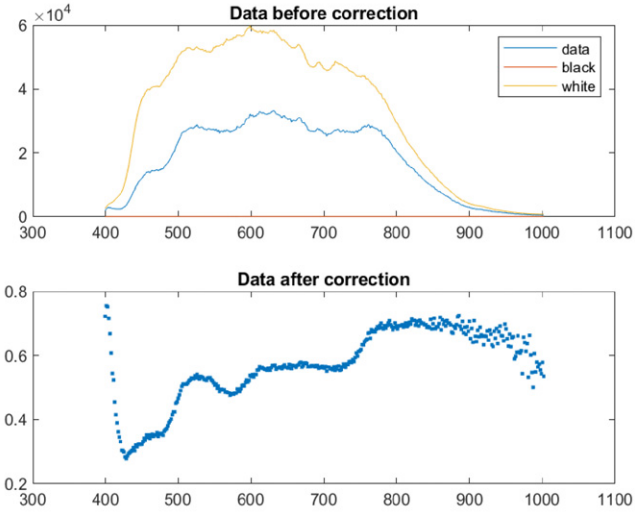

Figure 3. Data correction example.

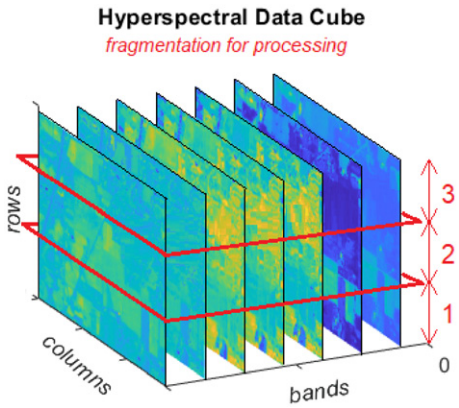

Figure 4. Segmentation of the hypercube into three chunks. Modification of Fig. 1.

3.3 *Data Correction and Hypercube Fragmentation*

First, calibration of the (raw) data extracted form the hyperspectral camera is performed. To do so, dark reference must be subtracted from raw data. The result of this operation is then normalised using the white reference. A one-pixel example of this correction is shown on Figure 3. Let us note that this action is not required in satellite imaging, but becomes very important in proximal hyperspectral imaging.

After data is corrected, the hyperspectral cube is split into equal sized chunks (cf. Figure 4) so that we can optimize computing with the different sections of the cube.

Once all chunks have been processed, they will be reordered to form the final hyperspectral image.

Note that we utilized file fragmentation as a means to enhance computing time, as parallel computing and threading did not yield the desired results. Specifically, employing parallel loops proved unsuccessful in constructing the hypercube accurately and often resulted in crashes or overwriting. Consequently, we made the decision to forgo parallel computing and instead relied on the fragmentation of the cube, which significantly reduced processing time.

Table I compares the execution time of different fragmentations of the cube. The same raw hyperspectral file (3,11 GB) was processed from 400 to 780 nm, with a step of 5 nm. We ran the software on a personal computer (Intel(R)

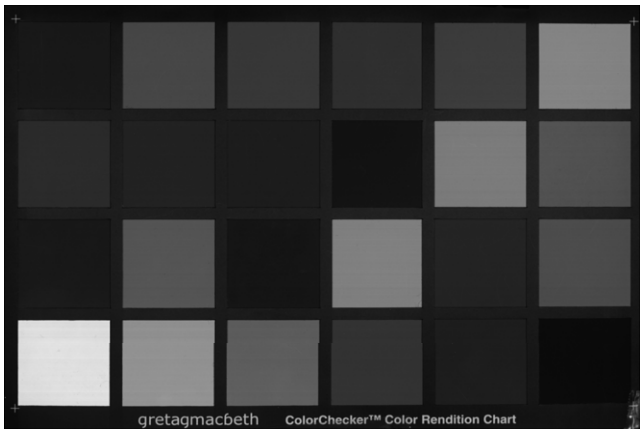

Figure 5. Photo at 525 nm under illuminnat D50.

Figure 6. Final photo of hyperspectral camera, all wavelengths (380–780 nm) integrated under illuminant D50.

Table I. "Performance Comparison: Fragmentation of Hyperspectral Cube versus Execution Time. Without file fragmentation (1 single file), our tool experiences crashes when processing heavy files on personal computers. File fragmentation was implemented to enhance stability, reduce processing time, and prevent crashes".

Number of pieces	Execution time (s)
1	Crash (out of memory)
2	194
3	89
4	75
5	73
Automatic (6)	74
8	76

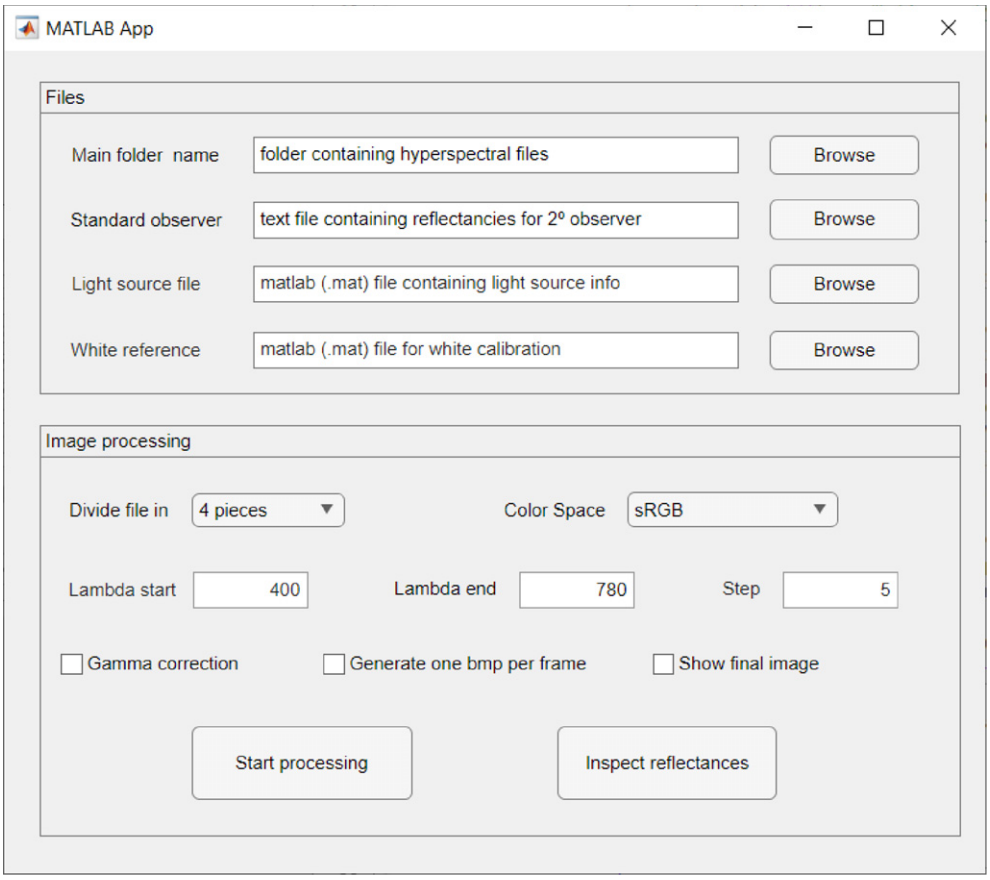

Figure 7. GUI main window.

Core(TM) i7-8750H CPU @ 2.20 GHz 2.21 GHz, installed RAM 16 GB) as any of our fellow researches in other fields might use.

3.4 *Tristimuli Calculation*

The calculation of the tristimuli T integrates the light source power spectral density, the reflectance of the object (hyperspectral image captured by the camera), and the standard observer response, along all the bands.

$$T = \int s(\lambda) \cdot r(\lambda) \cdot x(\lambda)\, d\lambda. \tag{1}$$

In Eq. (1), $s(\lambda)$ represents the light source for a certain wavelength λ. All the same, $r(\lambda)$ stands for the raw hyperspectral data, and $x(\lambda)$ for the standard observer. Note that the integration is approximated in the software by a Riemann sum.

3.5 *Color Space Transformation*

We have named "color space transformation" to the adaptation of the image to the device that will be used to visualize the final image. This transformation consists of a matrix multiplication (Eq. (2) gives the matrix example for a RGB screen) followed by the corresponding gamma transformation [5].

$$RGB = \begin{pmatrix} 2.19 & -0.69 & -0.32 \\ -0.87 & 1.80 & 0.02 \\ 0.04 & -0.086 & 0.88 \end{pmatrix}. \tag{2}$$

Numerous matrices can be used for different devices, which allows the hyperspectral image to be seen more accurately. Our tool includes RGB transformation for regular screen and HTC transformation for visualization using HTC-Vive headset.

After transformation of all the cube parts, the image can be reconstructed and saved in an image file. Two image files will be created by the software:

4. EXAMPLE OF IMAGE PROCESSING

A color checker was photographed using a pco.edge 4.2 hyperspectral camera, and the three ENVI files (raw data, dark reference and white reference) were produced with their corresponding headers. The images between 380 and 780 nanometers were processed using CIE D50 illuminant, dividing the cube into 5 equal pieces. The software generated a single image for each wavelength (see Figure 5 for 525 nanometers) and a final image with all wavelengths integrated (Figure 6).

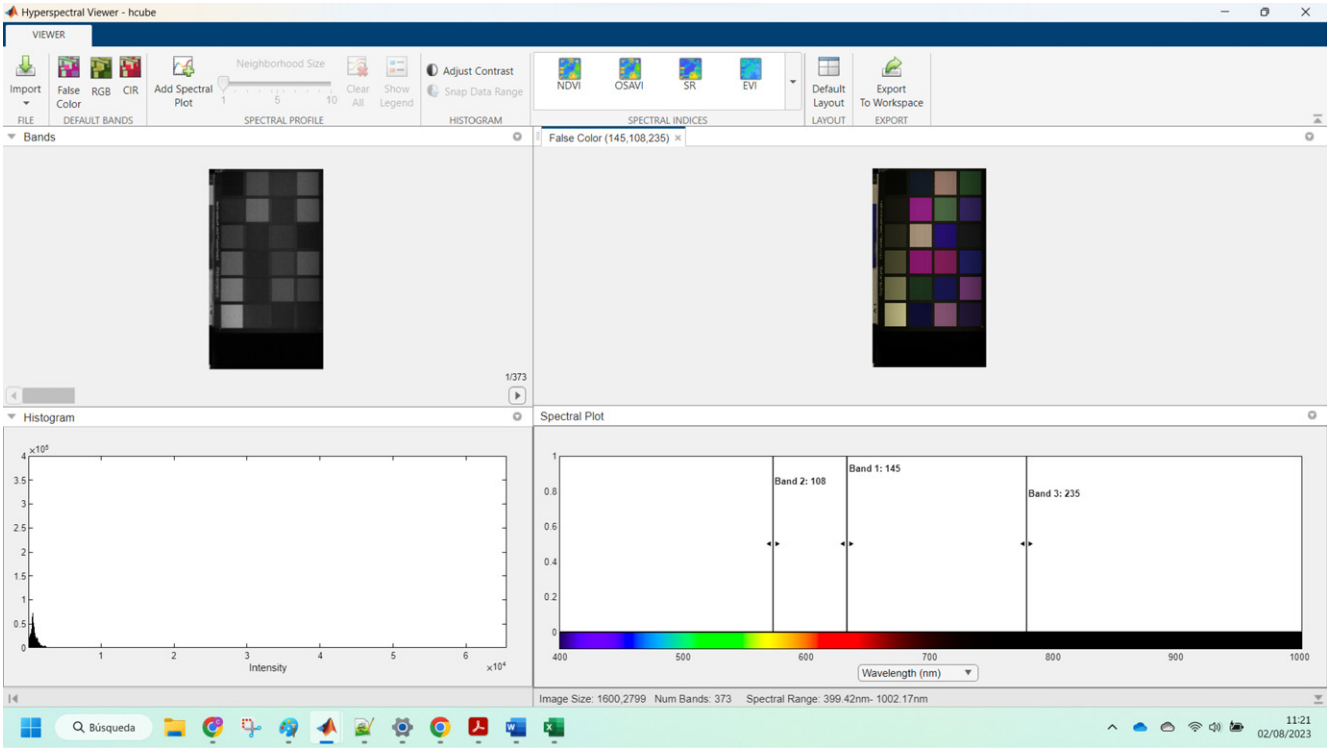

Figure 8. MATLAB tool (Hyperspectral viewer). While it offers a handy and easy-to-use interface, it may be challenging for untrained users, and it requires a MATLAB license.

5. GRAPHICAL USER INTERFACE

A GUI was implemented to enhance user experience (see Figure 7). Any person with basic computer knowledge can study the images captured by the hyperspectral camera.

Our tool offers the advantage of simplicity and does not require a MATLAB license. However, it is worth noting that while the MATLAB is able to process the entire raw file in 242 s, our tool experiences stability issues when processing it, leading to occasional crashes. Despite this limitation, we have achieved significant improvements in performance using file fragmentation (see Table I). Additionally, it is important to consider that image processing in MATLAB may render the same personal computer unresponsive during its execution. As for the interface, MATLAB provides a GUI that could pose challenges for users without prior training (see Figure 8).

The first use of the image processing is to generate a hyperspectral high-quality image (PNG); but there is much more to hyperspectral images. The GUI allows the user to generate an image (BMP) per frame (i.e., one image per band), which permits to detect nuances in color at a simple glance. Moreover, the software allows to inspect the final image pixel-by-pixel, as shown in Figure 9.

6. CONCLUSIONS

Proximal hyperspectral imaging uses brand new technology to obtain extremely accurate images, but the data extraction and handling can be time consuming and difficult to understand. We developed an algorithm to reduce not only processing time but also computing power; using this software, a simple PC is able to process images that would normally need a more powerful processor. It is coded in M language (MATLAB programming language) but does not require a license. In the study of hyperspectral imaging, we had requests from our colleagues in other departments with little computer imaging knowledge, which led us to the development of the GUI. We offer an easy-to-use, handy software tool for anyone. It goes from generating a hyperspectral image to pixel inspection or single wavelength photograph; from a novice to an advanced user, this is a tool that we hope can help making proximal hyperspectral images more accessible.

7. SUPPLEMENTARY MATERIAL

The software is available online at https://github.com/OrionMerida/TratamientoImagen.git.

8. FUNDING

This work was supported by the grant IB20094 of the Regional Government of the Junta de Extremadura and partially financed by the European Regional Development Fund.

9. CONFLICTS OF INTEREST

The authors declare no conflict of interest.

ACKNOWLEDGMENT

We would like to thank the Institute of Archaeology in Merida (Extremadura, Spain) for letting us know how helpful a hyperspectral processing tool would be.

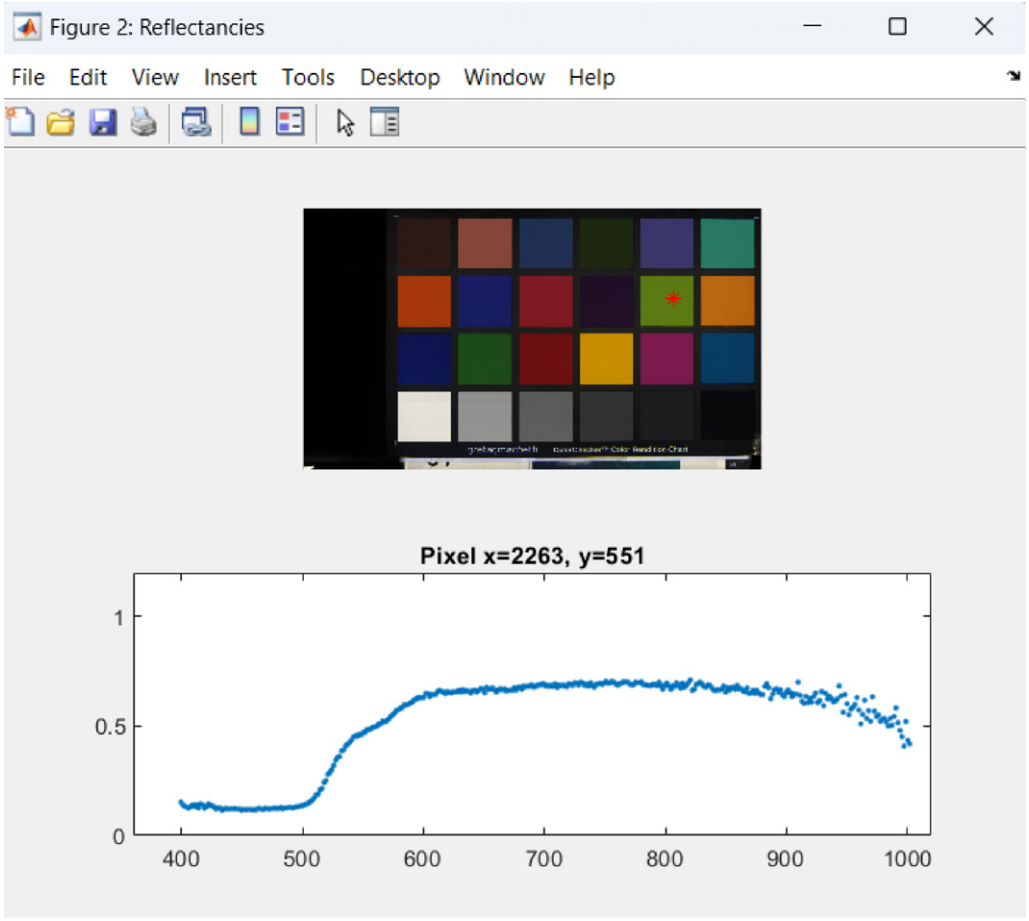

Figure 9. Reflectance Inspection: final hyperspectral image with the selected pixel and reflectance of that pixel across all bands.

REFERENCES

[1] J. M. Amigo, H. Babamoradi, and S. Elcoroaristizabal, "Hyperspectral image analysis. A tutorial," Anal. Chim. Acta **896** (2015).

[2] D. H. Foster and K. Amano, "Hyperspectral imaging in color vision research: tutorial," J. Opt. Soc. Am. A **36** (2019).

[3] Matlab Documentation. Available at: https://es.mathworks.com/help/images/getting-started-withhyperspectral-image-analysis.html. Accessed 04/17/2023.

[4] ISO 11664-2:2007(E)/CIE S 014-2/E:2006 Colorimetry — Part 2: CIE standard illuminants for colorimetry.

[5] CIE 015:2018 Colorimetry, 4th ed. DOI: 10.25039/TR.015.2018.