Analysis and project requirements

Team 7RS

2022-11-10

Abstract

Whaaale is a hyperspectral data viewer. It allows displaying images in monochrome and fake-coloured RGB modes. Analysis tools include spectral curves and similar pixel search.

Contents

Analysis

Whaaale is a program dedicated to displaying and studying spectral images. It offers several functionalities that can be divided into displaying an image of a loaded file and displaying properties of selected pixel or region in form of spectral curves. Program is easy to use so that user can work with it without any training.

Use case diagram

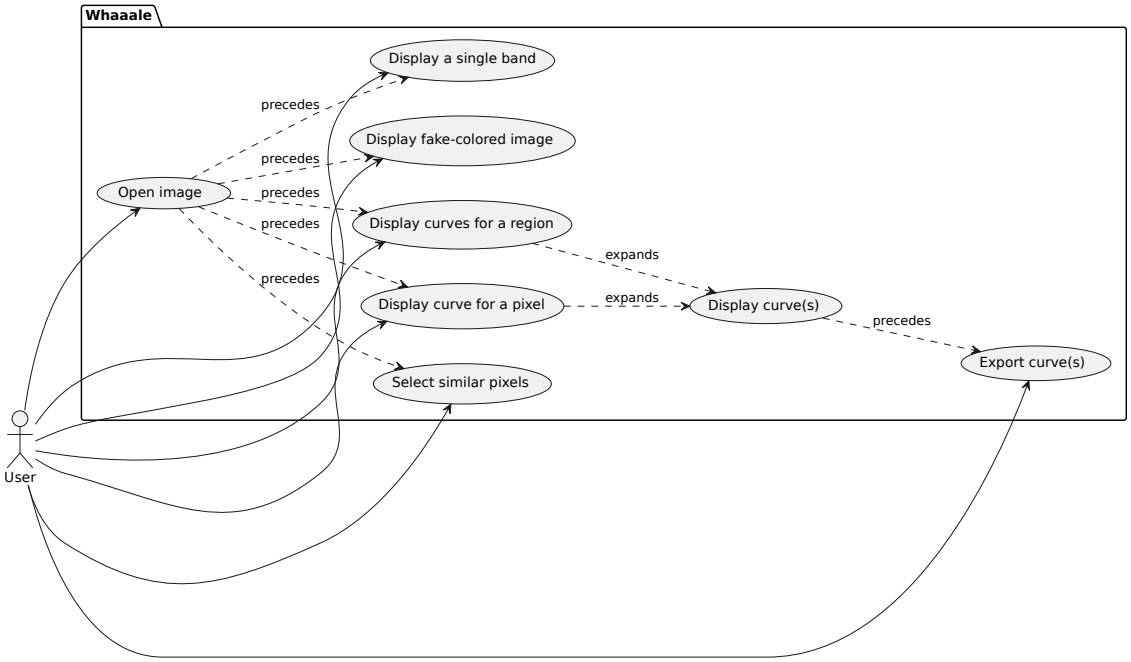

Figure 1: Use case diagam

1 Functional requirements

1.1 Loading files

Description: Loading hyperspectral image data from a file selected by the user.

Input data: Three-dimensional matrix (height x width x bands) with an optional list of wavelengths.

Input data source: MATLAB `.mat` file or ENVI `.hdr` labelled raster files (data and header).

Result: Data and labels are loaded into memory, preview is updated with freshly loaded data.

Pre-condition: Data file and in case of ENVI files its header exist, are not malformed and contain proper data.

Post-condition: New image is available for processing and displaying.

Side effects: Switch display mode to monochromatic and first band.

Reason: Function handles data loading.

1.2 Displaying a single band

Description: Displaying monochromatic image by presenting a chosen band.

Input data: Three-dimensional matrix (height x width x bands), band number.

Input data source: Currently loaded file, band input.

Results: Monochromatic image of the selected band is displayed.

Pre-condition: File is loaded into memory.

Post-condition: Monochromatic image is displayed.

Side effects: Only one band is displayed.

Reason: Function displays an image that is comprehensible for human.

1.3 Displaying a fake-colored image

Description: Displaying fake-colored image by assigning selected bands to RGB channels.

Input data: Three-dimensional matrix (height x width x bands), bands for RGB channels.

Input data source: Currently loaded file, R, G and B band inputs.

Results: Fake-colored image is displayed.

Pre-condition: File is loaded into memory.

Post-condition: Fake-colored image is displayed.

Side effects: Displayed image is fake-colored and might be different for different choice of bands.

Reason: Function displays an image that is comprehensible for human.

1.4 Displaying spectral curve

Description: Displaying a line chart with information about intensity for different bands.

Input data: A vector with intensity values for each band or a group of vectors representing average, minimum, maximum and quartile intensity values for each band; wavelength labels for bands (if exist).

Input data source: Selected area or pixel; header data of loaded image

Result: Display a line chart with values form vector(s) and labelled on x-axis with wavelengths or sequential band numbers.

Pre-condition: A pixel or an area is selected.

Post-condition: Chart is updated.

Side effects: New chart is ready for export.

Reason: Function handles displaying spectral curves.

1.4.1 Selecting a point

Description: Selecting a new point as a source for spectral curve.

Input data: Pixel value

Input data source: Pixel clicked by the user

Result: Spectral curve is updated with the value of the pixel.

Pre-condition: File is loaded into memory.

Post-condition: Curve is updated.

Side effects: Previous curve data is unavailable.

Reason: Function handles selecting a pxiel from the image.

1.4.2 Selecting an area

Description: Selecting an area as a source for spectral curve.

Input data: A range of pixels

Input data source: Pixel from within an area selected by the user.

Result: Spectral curve is updated with the average, minimum, maximum and quartile values of pixels from the range.

Pre-condition: File is loaded into memory.

Post-condition: Curve is updated.

Side effects: Average, minimum, maximum and quartile values are calculated. Previous curve data is unavailable.

Reason: Function handles selecting an area from the image.

1.4.3 Exporting the curve

Description: Exporting displayed curve as a CSV or PNG file.

Input data: Target file path, file type, curve data

Input data source: User input in file picker, curve

Result: File specified by the user is populated with curve data in the selected format.

Pre-condition: Curve is displayed.

Post-condition: File is written.

Side effects: *none*

Reason: Function handles exporting curve data.

1.5 Selecting similar pixels

Description: Selecting similar pixels by choosing one pixel. All the similar pixels are selected according to the chosen tolerance value.

Input data: Three-dimensional matrix (height x width x bands), a pixel, tolerance value.

Input data source: Currently loaded file, selected pixel, tolerance input.

Results: All the pixels that fall within the tolerance range are selected.

Pre-condition: File is loaded into memory.

Post-condition: Only the pixels that are within the tolerance range are selected.

Side effects: Image is automatically displayed in a single band mode to make the selection clearly visible.

Reason: Function handles selecting similar pixels in an image.

2 Non-functional requirements

2.1 Delivery

All the functionalities are delivered properly according to the functional requirements on the fixed time.

2.2 Performance

All operations on images with dimensions 200x300 px and 224 bands should complete under 10s on x86 computers supporting AVX2.

2.3 Space

Program should be require no more than four times the size of uncompressed input file (considering files that are bigger than 500 MB).

2.4 Portability

Program must be available for Linux and Windows. It must be possible to run the program without installing it. Program should be distributed as a ZIP archive.
If possible without developer's license and code signing certificate, the program should be available for macOS.

2.5 Usability

All buttons with icons must have a label visible after hovering them with a mouse cursor.

2.6 Reliability

All errors during file I/O (loading images, exporting curves) must be caught and displayed in a dialog box.

2.7 Code organisation

Code must adhere to the following criteria:

2.7.1 Text file format

All text files must be UTF-8 encoded and saved with LF line endings.

2.7.2 Python source style

Source code files in Python must be formatted with black. Code is compatibile with PEP 8 standard.

2.8 Licensing

Code must be open source and licensed under GPLv3.

2.9 Interoperability

Program should be able to load images from different formats specified in the functional requirements in the loading files section.