Documentation for Helen Davies's Multispectral Image Processing Script

Overview

This script is designed to process multispectral image data by applying various image analysis techniques such as Principal Component Analysis (PCA), Independent Component Analysis (ICA), image ratio calculations, Gaussian blurring, and a placeholder for Minimum Noise Fraction (MNF) transformation. It provides functionality to load image data from files, perform transformations, and save the results.

Requirements

To run this script, you need several Python libraries installed, including:
- **NumPy**: For array manipulation.
- **SciKit-Image (skimage)**: For image reading and applying Gaussian blur.
- **SciKit-Learn**: For PCA and ICA computations.
- **Rasterio**: For handling geospatial raster data.
- **Imageio**: For reading and writing images.

Installation of Libraries

Before running the script, ensure that you have the necessary Python libraries installed. You can install them using pip. Run the following commands in your command line:

```bash
pip install numpy
pip install scikit-image
pip install scikit-learn
pip install rasterio
pip install imageio
```

Functions in the Script

1. load_multispectral_image_bands(input_folder)
   - Purpose: Loads all TIFF images from the specified folder and stacks them into a single NumPy array.
   - Parameters :
     - `input_folder`: Path to the folder containing the TIFF images.
   - Returns : A 3D NumPy array where each slice along the first axis corresponds to a different image band.

2. perform_pca(image)
   - Purpose : Applies PCA to decompose the image into principal components.
   - Parameters :
     - `image`: A 3D NumPy array of image data.
   - Returns : Image transformed into its principal components.

3. perform_ica(image)
   - Purpose : Applies ICA to separate the image into independent components.
   - Parameters :

  - `image`: A 3D NumPy array of image data.
- Returns : Image transformed into its independent components.

4. create_image_ratio(image, band1, band2)
- Purpose : Calculates the ratio between two specified bands of an image.
- Parameters :
  - `image`: A 3D NumPy array of image data.
  - `band1`: Index of the numerator band.
  - `band2`: Index of the denominator band.
- Returns : A 2D array representing the ratio of the specified bands.

5. apply_gaussian_blur(image, sigma=1)
- Purpose : Applies Gaussian blur to the image.
- Parameters :
  - `image`: A 2D NumPy array of image data.
  - `sigma`: Standard deviation of the Gaussian kernel.
- Returns : Blurred image.

6. perform_mnf(image)
- Purpose : Placeholder for Minimum Noise Fraction transformation. Currently returns the input image unchanged.
- Parameters :
  - `image`: A 3D NumPy array of image data.
- Returns : Unchanged image (placeholder functionality).

7. save_image(image, output_folder, file_name, process_name)
- Purpose : Saves the image as a TIFF file with a descriptive filename indicating the process applied.
- Parameters :
  - `image`: Image data to save.
  - `output_folder`: Directory to save the image.
  - `file_name`: Base name for the file.
  - `process_name`: Description of the process applied to the image.
- Effects : Creates a TIFF file in the specified directory.

8. process_images(input_folder, output_folder)
- Purpose : Manages the workflow of loading images, processing them with various techniques, and saving the results.
- Parameters :
  - `input_folder`: Directory containing input TIFF images.
  - `output_folder`: Directory where processed images will be saved.

Usage Example
```python
# Define paths to your input and output directories
input_folder = 'path_to_your_input_folder'
```

output_folder = 'path_to_your_output_folder'

Important Notes

- Ensure that the input directory contains only TIFF files and that they are named consistently to maintain correct band order.
- This script does not handle exceptions extensively; additional error handling may be required for production use.
- The script assumes that image data is stored in a format compatible with the processing functions used (e.g., images should be appropriately scaled and formatted).

By following this documentation, you should be able to install necessary dependencies, understand the functions within the script, and run the script to process multispectral images effectively.