EUROPEAN SPACE AGENCY

ESA Reference: ESA AO/1-11038/21/I-DT

ESA Contract Number: 4000139605/22/I-DT

Project Afri4Cast - Food Security and Safety In Africa

WP200

D8.1 Associated Documentation of Processor/Toolbox/Software

Submitted by:

In collaboration with:

23 September 2024

Document Information

ESA Contract Number	4000139605/22/I-DT	**Acronym**	AFRI4CAST
Full Title	EO AFRICA EXPLORERS – EXPRO+ Food Security and Safety In Africa		
Start Date	26th October 2022	**Duration**	24 months
Project URL	www.afri4cast.org		
Deliverable	D8 – Associated Documentation of Processor/Toolbox/Software		
Work Package	WP200 – Prototype test and Outreach		
Date of Delivery	**Contractual**	**Actual**	Contractual
Nature	Report	**Dissemination Level**	Confidential
Lead Beneficiary	AgroApps		
Responsible Author	Evangelos Oikonomopoulos		
Contributions from	SIA		

Document History

Version	Issue Date	Pages	Changes
V1	26/04/2024	32	
V2	23/09/2024	32	No changes applied

Disclaimer

This document and its content reflect only the author's view, therefore ESA is not responsible for any use that may be made of the information it contains!

Contents

1. Introduction

The aim of this document is to provide a comprehensive overview of the tools and software that are essential to AFRI4Cast's methodologies and processes. Firstly, the documentation presents in detail the tools, packages and software that were used in the thermal data fusion steps, such as the thermal images acquisition, preprocessing, the diurnal temperature cycle modeling and, lastly, the realization of the spatiotemporal thermal data fusion. It also provides throughout descriptions regarding the tool, packages and software that were utilized in each individual step for the implementation of the hyperspectral/ multispectral data fusion, i.e. hyperspectral/ multispectral images acquisition, images preprocessing and coregistration and finally the data fusion procedure. Lastly, the related tools and packages that were exploited for the inversion of RTMs for biophysical parameters process as well as for the extraction of data for the crop growth and rust prediction models are mentioned.

2. Terms, Definitions and Abbreviation Terms

Table of Acronyms

Acronym	Full Term
AOI	Area of Interest
AFRI1600	Aerosol free vegetation index 1600
API	Application Programming Interface
ARVI	Atmospherically Resistant Vegetation Index
ARVI2	Atmospherically Resistant Vegetation Index 2
AVI	Ashburn Vegetation Index
AppEEARS	Application for Extracting and Exploring Analysis Ready Samples
AROSICS	Automated and Robust Open-Source Image Co-Registration Software
ASI	Agenzia Spaziale Italiana (Italian Space Agency)
BWDRVI	Blue-wide dynamic range vegetation index
CC	Canopy Cover
CDO	Climate Data Operators
CDS	Climate Data Store
CIgreen	Chlorophyll Index Green
CI	Coloration Index
DTC	Diurnal Temperature Cycle
DVIMSS	Differenced Vegetation Index MSS
ECOSTRESS	ECOsystem Spaceborne Thermal Radiometer Experiment on Space Station
EODAG	Earth Observation Data Access Gateway
EVI	Enhanced Vegetation Index
EVI2	Enhanced Vegetation Index 2
FE2	Ferric iron, Fe2+
FE3	Ferric iron, Fe3+
FWHM	Full width half maximums
GARI	Green atmospherically resistant vegetation index
GBNDVI	Green-Blue NDVI
GEE	Google Earth Engine
GLP	Gaussian Laplacian Pyramid
GNDVI	Green Normalized Difference Vegetation Index
GOSAVI	Green Optimized Soil Adjusted Vegetation Index
GRNDVI	Green-Red NDVI
GreenWDRVI	Green Wide Dynamic Range Vegetation Index
GSAVI	Green Soil Adjusted Vegetation Index
GUI	Graphical User Interface
GVMI	Global Vegetation Moisture Index

HS	Hyperspectral
IVI	Ideal vegetation index
LAI	Leaf area index
LST	Land Surface Temperature
MODIS	Moderate Resolution Imaging Spectroradiometer
MS	Multispectral
MTF	Modulation Transfer Function
MSI	MultiSpectral Instrument
MVI	Mid-infrared vegetation index
NDVI	Normalized Difference Vegetation Index
Norm G	Norm Green
Norm NIR	Norm NIR
Norm R	Norm Red
PPR	Normalized Difference 550/450 Plant pigment ratio
PRISMA	PRecursore IperSpettrale della Missione Applicativa
PVR	Normalized Difference 550/450 Plant pigment ratio
RTM	Radiative Transfer Model
SIWSI	Normalized Difference 860/1640
SLAVI	Specific Leaf Area Vegetation Index
SR	Super Resolution
SRF	Spectral Response Function
SWIR	Shortwave Infrared
VARIgreen	Visible Atmospherically Resistant Index Green
VNIR	Visible and Near-Infrared
WDVI	Weighted Difference Vegetation Index
WRDVI	Wide Dynamic Range Vegetation Index

3. Thermal Data Fusion

3.1. Thermal Image Acquisition

This chapter includes the description of the tools used within AFRI4Cast for the acquisition of thermal images. The tools concern the various APIs that are used to retrieve the thermal data products or their metadata, which aid in the implementation of the data processing pipelines.

AppEEARSAPI

Within AFRI4Cast, AppEEARSAPI, an Application Programming Interface that allows users to interact with AppEEARSdata archives, was used in order to acquire ECOSTRESS and MODIS Terra/Aqua LST data products through Python scripting.

AρρEEARS, the Application for Extracting and Exploring Analysis Ready Samples, allows users to easily access and obtain geographic data from various Earth observation data sources. AppEEARSprovides the ability to subset geospatial satellite datasets using geographical, temporal, and dataset layer parameters (https://appeears.earthdatacloud.nasa.gov/api/). Two different types of sample requests are available: point samples with the use of geographic coordinates and area samples for spatial areas using vector polygons, the latter of which is used in AFRI4Cast.

The custom Python script that was developed for realizing API requests and interacting with the AppEEARSdata archive was executed via a Python virtual environment. Firstly, the requirements (packages) demanded by the AppEEARSAPI Python script were installed in this virtual environment by a “requirements.txt” file. These requirements included the following packages:

- **geopandas**, Version 0.12.2 : A package to work with geospatial data in Python.
- **requests**, Version 2.28.2 : A package that allows users to easily send HTTP/1.1 requests
- **Datetime**, Version 5.1 : This module provides classes for manipulating dates and times.
- **python-dotenv**, Version 1.0.0 : Reads key-value pairs from a .env file and set them as environment variables
- **argparse**, Version 1.4.0: A parser for command-line options and arguments

"**Json**" package was also utilized in order to import the vector file of the Area of Interest (AOI).

Lastly, an .env file was used, comprising a credentials file, in which the NASA Earthdata login username and password were stored and provided to the script.

The user was prompted to provide the following inputs in the .env file as well as in the command line arguments:

- **Username**: NASA Earthdata Login Username in .env file
- **Password**: NASA Earthdata Login Password in .env file

Command-line arguments:

- **-d**: The desired directory for the outputs
- **-p**: Product name & version in the format ProductName.Version (e.g.MOD11A1.061)
- **-l:** The desired layers for the chosen product (e.g.LST_Day_1km, LST_Night_1km)
- **-g**: The directory where the shape geojson file of the ROI is stored.
- **-t**: Preferred task name
- **-sd**: Start Date in MM-DD-YYYY format
- **-ed**: End Date in MM-DD-YYYY format

The script was executed on the command-line via the following command:

```
python AppEEARS_API_Area_Request.py -d /path/to/target/directory -p "ProductName.Version" -l "Layer names for the product separated by commas" -g /Path/to/ROI/geojson/file -t "TaskName" -sd "Start date in MM-DD-YYYY format" -ed "End date in MM-DD-YYYY format"
```

EODAG

Within AFRI4Cast, EODAG, the Earth Observation Data and Access Gateway, was used to explore and acquire Copernicus Sentinel-3 SLSTR Level-2 LST data products through custom Python scripting.

EODAG is a command-line tool as well as a Python package that simplifies the process of accessing and obtaining Earth observation data. EODAG makes it easier to search for, access and download satellite imagery, while it also provides a unified API for accessing Earth observation data regardless of the data provider (https://eodag.readthedocs.io/en/stable/). In the context of AFRI4Cast, a custom developed Python script uses the EODAG package and it is set up to work with CreoDIAS, Copernicus Dataspace Ecosystem, and ONDA DIAS API.

The custom Python script that was developed for accessing and retrieving Sentinel-3 SLSTR Level-2 LST data with the use of EODAG package was executed via a Python virtual environment. The requirements (packages) requested by this Python script were installed in the virtual environment by a "requirements.txt" file and they included the following:

- **eodag**
- **python-dotenv**, Version 1.0.0 : Reads key-value pairs from a .env file and set them as environment variables
- **argparse**, Version 1.4.0: A parser for command-line options and arguments

An .env credentials file was also used in order to store and provide the script with the login credentials (username and password) of the defined providers (CreoDIAS, Copernicus Dataspace Ecosystem and ONDA DIAS).

The user was prompted to provide the following inputs as command-line arguments:

- **Workspace directory (--workspace):** The desired workspace directory where the products will be saved
- **Extent (--extent):** The WKT string in the format Polygon (())
- **Start Date (--start_date):** Start Date in YYYY-MM-DD format

- **End Date (--end_date):** End Date in YYYY-MM-DD format

The script was executed on the command-line via the following command:

```
python EODAG_S3_L2LST.py --workspace "path/to/workspace/directory" --extent "WKT String" --start_date "YYYY-MM-DD" --end_date "YYYY-MM-DD"
```

Climate Data Store API

Copernicus Climate Data Store (CDS) API is a service that allows access to CDS data archives (https://cds.climate.copernicus.eu/api-how-to). This collection includes the ERA5-Land REA data product, which was used in AFRI4Cast's pipelines. The product layer "Skin temperature", which describes the temperature of the Earth's surface was requested by defining:

- the AOI as a bounding polygon
- the product type
- the product variable
- the time period (year, month, day)
- the requested hours
- the format (GRIB)

3.2. Thermal Image Processing

The corresponding tools and software that were utilized in order to develop the algorithm codes for the implementation of the thermal images preprocessing pipelines are presented in this chapter. For detailed information regarding the thermal images preprocessing approach for each data product as well as the corresponding outline, please refer to the Deliverable D7.

MODIS Terra / Aqua LST and View Time

The preprocessing steps for MODIS Terra/ Aqua Day/ Night LST and View Time layers included the cloud masking, the data scaling and the spatial resampling. In order to accomplish these steps, a custom Python code was developed. The cloud masking was based on the Quality Assurance(QA) bit flags provided by the Quality Control (QC) layers along with each MODIS Terra/ Aqua Day/ Night LST data product and a bitwise extraction approach. The data scaling was performed within this Python code as well. Lastly, the spatial resampling to the ECOSTRESS LST grid system was realized on SAGA GIS via the nearest neighbor method.

The Python script for the implementation of cloud masking and data scaling was executed in a Python virtual environment, in which the demanded packages were firstly installed by a "requirements.txt" file. These requirements included:

- **rasterio**, Version 1.3.9 : Reads and writes gridded or raster datasets. It is used with Numpy and it works with Python 3.8+, Numpy 1.18+, and GDAL 3.1+ (https://rasterio.readthedocs.io/en/stable/) .
- **numpy**, Version 1.26.1 : A fundamental package for array computing (https://numpy.org/doc/stable/user/index.html#user)
- **argparse**, Version 1.4.0: A parser for command-line options and arguments (https://docs.python.org/3/library/argparse.html)

The user was prompted to provide the following inputs as command-line arguments:

- **--qc_file:** Path to the MODIS Terra/ Aqua Day/ Night QC layer for a specific day
- **--lst_file:** Path to the corresponding MODIS Terra/ Aqua Day/ Night LST or View Time layer for the same day
- **--output_path:** The desired path in which the masked and scaled output (MODIS Terra/ Aqua Day/ Night LST or View Time) (Data type: Float32, No data value: NaN) will be stored

In order for the cloud masking to be performed, the user should provide the appropriate decimal values that correspond to the desired binary representation of each bit pair and their key values. The QA bit

flags and their key values for MOD11A1.061 and MYD11A1.061 QC layers are provided in Deliverable D7. So, in order to conduct a proper cloud masking, all values bigger than the binary 10 in the bit pair 1 & 0 should be excluded. Thus, the bit pairs and their key values were formed as follows:

```
qa_mask = bitwiseExtract(qc, 0, 1) <= 1

data_quality_mask = bitwiseExtract(qc, 2, 3) == 0

emis_error_mask = bitwiseExtract(qc, 4, 5) == 0

lst_error_mask = bitwiseExtract(qc, 6, 7) == 0
```

Lastly, the script was executed on the command-line via the following command:

```
python QualityMasking.py --qc_file /path/to/QC_Layer/file.tif --lst_file /path/to/LST_Layer/file.tif --output_path /path/to/save/the/masked/MaskedOutput.tif
```

ECOSTRESS LST

Similarly, to MODIS Terra/ Aqua Day/ Night LST and View Time cloud masking, a custom Python code for the cloud masking and the data scaling of ECOSTRESS LST layers was developed as well. The cloud masking relied on the QA bit flags provided by the QC layers that come along with the ECOSTRESS LST data product and a bitwise extraction approach. The data scaling was also realized by the same Python script.

The Python script for the implementation of cloud masking and data scaling was executed in a Python virtual environment, in which the demanded requirements were firstly installed by a "requirements.txt" file. These packages included:

- **rasterio**, Version 1.3.9: Reads and writes gridded or raster datasets. It is used with Numpy and it works with Python 3.8+, Numpy 1.18+, and GDAL 3.1+ (https://rasterio.readthedocs.io/en/stable/).

- **numpy**, Version 1.26.1: A fundamental package for array computing (https://numpy.org/doc/stable/user/index.html#user)
- **argparse**, Version 1.4.0: A parser for command-line options and arguments (https://docs.python.org/3/library/argparse.html)

The user was prompted to provide the following inputs as command-line arguments:

- **--qc_file**: Path to the ECO2LSTE.001 QC layer for a specific day
- **--lst_file**: Path to the corresponding ECO2LSTE.001 LST layer for the same day
- **--output_path_LST**: The desired path in which the masked and scaled LST output (Data type: Float32, No data value: NaN) will be stored
- **--output_path_BM**: The desired path in which the binary mask will be stored (Data type: Uint8, No data value: 255)

In order for the cloud masking to be performed, the user should provide the appropriate decimal values that correspond to the desired binary representation of each bit pair and their key values. The QA bit flags and their key values for ECO2LSTE.001 QC layers are provided in Deliverable D7. Thus, in order to conduct a proper cloud masking, all values different than 0 in the Bit pair 1 & 0 should be excluded.

Lastly, the script was executed on the command-line via the following command:

```
python QualityMasking_ECOSTRESS.py --qc_file /path/to/ECOSTRESS QC_Layer/file.tif --lst_file /path/to/ECOSTRESS LST_Layer/file.tif --output_path_LST /path & filename/to/save/the/masked LST/MaskedOutput.tif --output_path_BM /path & filename/to/save/the/binary mask/BM.tif
```

Sentinel-3 SLSTR L2 LST

As already mentioned in the Deliverable D7, Chapter 3.5., the preprocessing steps for Sentinel-3 SLSTR Level-2 LST data products include the data clipping, the reprojection and the cloud masking of the data. A preprocessing graph through ESA'S SNAP software was developed in order to fulfill these steps. Specifically, the cloud masking was performed by exploiting the bit 14 (summary_cloud) of the confidence_in.nc layer that accompanies the Sentinel-3 SLSTR L2 LST data product.

Parameters to be set by the user for the execution of the S3 Quality – Cloud masking SNAP GPT:

- **-PInput**: File path to the input Sentinel-3 LST Image metadata (Scene Folder/xfdumanifest.xml)
- **-Pwkt**: Spatial subset polygon boundaries in WKT format
- **-POutput_LST**: File path to the LST output & File Name

The script was executed on the command-line via the following command:

```
gpt9 GPT_S3_LST.xml \
-PInput="Scene Folder.SEN3/xfdumanifest.xml" \
-Pwkt="wkt" \
-POutput_LST="LST Output Path & File name.tif"
```

Moreover, in some cases the extent of the study area intersects with two or more Sentinel-3 grid tiles. Therefore, a mosaicking process of the different Sentinel-3 LST masked images took place in this case. Lastly, the spatial resampling of the masked and mosaicked Sentinel-3 LST image to the MODIS LST grid system was conducted on SAGA GIS.

ERA5-Land REA Skin Temperature

ERA5-Land REA Skin Temperature variable layers were preprocessed by utilizing Unidata CDO software as well as R packages.

The Climate Data Operator (CDO) software is a collection of many operators for standard processing of climate and forecast model data. Simple statistical and arithmetic functions, data selection and subsampling tools and spatial interpolation are included in the operators. CDO was developed in order to provide in one package the same set of processing functions for GRIB as well as NetCDF datasets.

https://code.mpimet.mpg.de/projects/cdo/embedded/index.html

Within the ERA5-Land REA Skin temperature layers preprocessing, CDO was used for the following preprocessing steps:

- Inspect GRIB/GRIdded Binary file and translate format to netCDF. Create a relative time axis.
- Concatenate 2 days of data, to capture the full DTC
- Perform linear interpolation between timesteps
- Calculate Statistics, from which T Maximum and T Minimum layers of the DTC are computed
- Calculate Argmax of Temperature (tm), which is the actual time where T is maximum

For some of the above mentioned steps, various R packages were used as well:

- **"zoo"** package was used along with CDO in order to perform the linear interpolation between timesteps. "Zoo" package offers methods for totally ordered indexed observations, particularly targeting an irregular time series of numeric vectors/matrices (https://cran.r-project.org/web/packages/zoo/zoo.pdf) .
- **"dplyr"** package was used together with CDO for the calculation of Argmax of Temperature (tm). "dplyr" package aims at data manipulation by supporting common data manipulation processes (https://cran.r-project.org/web/packages/dplyr/dplyr.pdf) . For the same preprocessing step, "raster" package was used as well, which supports the reading, writing, manipulating and modeling of spatial data (https://cran.r-project.org/web/packages/raster/raster.pdf) .

The last preprocessing step, which constitutes the conversion of time values from Greenwich UTC time zone to local solar time, was fully conducted with the use of "raster" and "solartime" R packages. As mentioned above, "raster" is a package for geospatial data analysis and modeling, while "solartime" offers utilities for dealing with solar time (https://cran.r-project.org/web/packages/solartime/solartime.pdf) .

3.3. Diurnal Temperature Cycle (DTC) Modeling

Within AFRI4Cast, a custom Python code was developed for the fitting of MODIS data to a Diurnal Temperature Cycle (DTC) mode, the extraction of the parameters that optimize it and its application in order to calculate LST at the time of ECOSTRESS data acquisition. The Python code was developed within a Jupyter notebook.

The code inserts ERA5 Land data already downloaded and the user selects the values for the day of year, the delta T and the max iterations. The script then calculates Ta, To, ts and tm from the reanalysis data and it reads the four MODIS Terra/ Aqua Day/Night files and converts them to NetCDF format, as well. Afterwards, for each latitude - longitude pair, it receives the four LST values and performs the Levenberg-Marquardt algorithm to predict the LST at the time the user defines. Lastly, the output of the script is a NetCDF file that contains the predictions.

The user inputs and the Python packages used in each section of the code are presented below:

- ***User inputs definition:***

User Inputs

- `path_to_data`: Directory containing NetCDF files.

- `name_date`: Date of MODIS data.

- `day_of_year`: Day of the year.

- `delta_T`: Delta T value.

- `max_iterations`: Maximum iterations for LM algorithm.

- `variable_name`: Name of variable inside the converted to NetCDF files.

- `variable_name2`: Name of variable inside reanalysis data.

- `time_1`: Time for predicting LST.

- `output_file`: Name of the output file.

- `file_path1`: Directory containing ERA5 Land data.

- ***Functions definition:***

Python Packages

- math: For mathematical functions and operations (https://docs.python.org/3/library/math.html) .

- ***Reanalysis data in a dictionary:***

Python Packages

- math: For mathematical functions and operations (https://docs.python.org/3/library/math.html) .

- netCDF4: Reading and writing netCDF files (https://unidata.github.io/netcdf4-python/)

- numpy: For array computing

- math: For mathematical functions and operations

- ***Conversion of MODIS tif files to NetCDF format:***

User Inputs

- `modis_data`: Directory containing MODIS data.

Python Packages

- **os:** For operating system dependent functionality.
- **glob**: For pathname pattern expansion (https://docs.python.org/3/library/glob.html) .
- **subprocess**: For setting up new processes (https://docs.python.org/3/library/subprocess.html) .

- **MODIS data in a dictionary:**

Python Packages

- numpy: For array computing
- glob: For pathname pattern expansion.
- netCDF4.dataset: Module for reading netCDF files

- **Make the predictions with Levenberg-Marquardr algorithm:**

Python Packages

- math: For mathematical functions and operations
- warnings: For handling warnings
- time: For timing the execution
- pandas: For data manipulation and analysis (https://pandas.pydata.org/docs/user_guide/index.html)
- numpy: For array computing
- matplotlib.pyplot: For plotting graphs (https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html)
- scipy.optimize.curve_fit: Uses nonlinear least squares to fit a function (https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html)

3.4. Spatiotemporal Thermal Data Fusion with Deep Learning (STFN)

For the implementation of the Spatiotemporal Thermal Data Fusion (STTFN) with deep learning, a custom Python code was developed and several Python packages were used for its realization. These packages included:

- **Tensorflow** : A software library for machine learning and artificial intelligence. It can be used in a variety of applications but its main focus is on deep neural networks training and inference (https://www.tensorflow.org/guide) .

- **numpy** : A fundamental package for array computing.

- **cv2** : The main module in OpenCV that allows access to an easy-to-use interface for working with image processing functions (https://pypi.org/project/opencv-python/) .

- **matplotlib.pyplot** : A collection of functions that make matplotlib work like MATLAB. pyplot functions for creating a figure or a plotting area within the figure, decorating the plot with labels etc are provided.

- **datetime** : A module that supplies classes for manipulating dates and times.

- **skimage** : An image processing toolbox in Python, also known as "scikit-image", which builds on numpy, scipy. ndimage and other libraries (https://scikit-image.org/docs/stable/) .

- **patchify** : Splits images into small patches with overlap based on a provided patch cell size and merges patches back into the original image as well (https://pypi.org/project/patchify/0) .

- **rasterio**: Reads and writes gridded or raster datasets. It is used with Numpy and it works with Python 3.8+, Numpy 1.18+, and GDAL 3.1+ .

- **sklearn.metrics** : A module of scikit Python machine learning library that provides metric functions for the assessment of prediction error. In the case of AFRI4Cast's STTFN, the mean squared error and the mean absolute error metric functions were used (https://scikit-learn.org/stable/modules/model_evaluation.html) .

4. Hyperspectral Data Fusion

4.1. Image Acquisition

The tools and platforms used within AFRI4Cast for the acquisition of hyperspectral and multispectral imagery in terms of the hyperspectral/ multispectral data fusion are presented in this chapter.

PRISMA L2D

Within AFRI4Cast, PRISMA Level 2D imagery acquisition was realized via ASI's PRISMA Platform (https://prisma.asi.it). In AFRI4Cast's case, PRISMA L2D imagery was not available in the archive for the study sites, so an acquisition request was placed. Specifically, the tool "Mission Planning Operation" was used. Through this tool, new acquisitions can be requested and obtained since this tool allows the users to place acquisition requests, which will be integrated into the mission's acquisition plan according to an agreed prioritization policy.

The bare minimum settings for the submission of a new PRISMA acquisition though the "Mission Planning Operation" tool are specified by:

1) the latitude and longitude of the image center (decimal degrees),
2) the maximum permitted percentage of cloud coverage (%)
3) the minimum and maximum Look Angle
4) the minimum and maximum Sun Zenith Angle
5) the type of product which shall be generated at image acquisition completion.

Sentinel-2 MSI L2A

The EODAG Python package was used for the retrieval of Sentinel-2 MSI Level 2A imagery. As the usage of this package was similar to the case of thermal data, for more information regarding the EODAG Python package please refer to the Chapter 3.1. on the section "EODAG".

4.2. Image Pre-Processing

4.2.1. Multispectral Data

Within AFRI4Cast, the main Sentinel-2 MSI band fusion part was realized with a hyper-sharpening adaptation of the Modulation Transfer Function - Gaussian Laplacian Pyramid (GLP), a method that was adopted due to its computational efficiency and its good performance on S2 data. The Python code, that was utilized and modified for the realization of this method, was found in the following GitHub repository:

https://github.com/codegaj/py_pansharpening/tree/master

The Python packages that take part in the code implementation are listed below:

- **cv2** : The main module in OpenCV that allows access to an easy-to-use interface for working with image processing functions.
- **numpy** : A fundamental package for array computing.
- **matplotlib.pyplot** : A collection of functions that make matplotlib work like MATLAB. pyplot functions for creating a figure or a plotting area within the figure, decorating the plot with labels etc are provided.
- **scipy**: A Python library that is mainly used for scientific computing. It provides modules for optimization, linear algebra, interpolation, signal and image processing (https://docs.scipy.org/doc/scipy/tutorial/index.html#user-guide) .
- **rasterio**: Reads and writes gridded or raster datasets. It is used with Numpy and it works with Python 3.8+, Numpy 1.18+, and GDAL 3.1+ .

4.2.2. Hyperspectral Data

Hyperspectral data preprocessing constituted an important part of the hyperspectral cube's preparation for fusion with the Sentinel-2 data. The preprocessing process included the elimination of the VNIR/ SWIR overlapping bands, the atmospheric water absorption bands as well as the bad (noisy) bands. For extensive information concerning the preprocessing of the selected PRISMA image, please refer to the Deliverable D7.

Within AFRI4Cast, PRISMA data were manually imported, examined and processed in EnMAP-Box 3. The EnMAP-Box constitutes an open source Python plugin for QGIS that visualizes and processes hyperspectral remote sensing raster data. In particular, it is designed to manage and process imaging spectroscopy data from the EnMAP sensor, but also other hyperspectral data products (https://enmap-box.readthedocs.io/en/latest/) .

The EnMAP-Box plugin combines in an efficient way the benefits of QGIS (e.g., for visualization), packages like GDAL as well as various Python libraries. The plugin offers: a) a graphical user interface (GUI) for hyperspectral data visualization and, for example, spectral library management, b) a collection of cutting-edge algorithms in order to process high dimensional spectral and temporal remote sensing data. As a result, the algorithms included in the EnMAP-Box are of high value for other hyperspectral Earth observation missions (e.g. PRISMA) as well.

4.2.3. Hyperspectral – Multispectral Image co-registration

PRISMA hyperspectral (HS) and Sentinel-2 multispectral (MS) images serve as input data in the hyperspectral/ multispectral data fusion procedure supporting the spatial enhancement of PRISMA hyperspectral bands. However, the geospatial misalignment between these datasets poses an implication in the HS/ MS image fusion process. In order to proceed and correct the spatial displacements between the HS and the MS image, an image co-registration approach was adopted.

In terms of AFRI4Cast, the co-registration approach was implemented with the use of AROSICS (Automated and Robust Open-Source Image Co-Registration Software). AROSICS is a powerful open-source image co-registration tool for multi-sensor satellite data. It is designed to detect and correct the local and global misalignments, which are common in satellite imagery, between two input images at a

subpixel level. AROSICS is available both as a command line tool and as a Python package (https://danschef.git-pages.gfz-potsdam.de/arosics/doc/about.html) . Moreover, it offers two co-registration modes: a) the local co-registration approach that considers the locally varying shifts and calculates a number of tie points inside the overlap area of the input images, and b) the global co-registration approach that computes the displacements within a small image subset and then a single displacement vector is utilized to shift the target image. In the case of AFRI4Cast, the local co-registration approach was chosen. For more detailed information regarding these two approaches, please refer to Deliverable D7.

In order to apply the local co-registration approach, the reference and the target images (inputs) were defined as inputs.

- **Reference Image**: Source path of reference image (The multispectral image, i.e., Sentinel-2)
- **Target Image**: Source path of image to be shifted (The hyperspectral image, i.e., PRISMA)

Moreover, the grid resolution and the matching window size arguments were utilized as tuning parameters. After several tests that were performed, these tuning parameters received the following values:

- **Grid resolution (grid_res)** (float): 25
- **Window size (window_size)** (Tuple[int, int]): (64,64)

Apart from the tuning parameters, the below mentioned arguments were also defined:

- **path_out** (str) – target path of the coregistered image
- **fmt_out** (str) – raster file format for output file. Can be any GDAL compatible raster file format (e.g. 'ENVI', 'GTIFF'; default: ENVI)
- **r_b4match** (int) – band of reference image to be used for matching
- **s_b4match** (int) – band of shift image to be used for matching

- **max_shift** (int) – maximum shift distance in reference image pixel units (default: 5 pixel)
- **align_grids** (bool) – True: align the input coordinate grid to the reference (does not affect the output pixel size as long as input and output pixel sizes are compatible
- **out_gsd** (float) – output pixel size in units of the reference coordinate system
- **v** (bool) – verbose mode

Therefore, in order to perform the local co-registration, the following commands were executed in Python console of a virtual environment, in which AROSICS Python package was already installed:

```
from arosics import COREG_LOCAL

# Set input images paths
im_reference = '/path/to/Reference/Multispectral/Image/S2.tif'
im_target = '/path/to/Target/Hyperspectral/Image/HS_Cube_26052020.tif'

# Set arguments
kwargs = {
  'grid_res'          : 25,
  'window_size'   : (64,64),
  'path_out'          : '/path/to/store/output/coregistered/image/cor.tif',
  'fmt_out'           : 'GTIFF',
  'r_b4match'       : 7,# We chose to work with the bands that belong to the NIR spectral range. These bands correspond to Band 45 of the HS image and Band 7 in the MS composite.
  's_b4match'       : 45,
  'max_shift'       : 15,
  'align_grids' : True,
  'out_gsd'           : [30,30],
  'v'         : True,
}

# Perform local coregistration
CRL = COREG_LOCAL(im_reference,im_target,**kwargs)
CRL.correct_shifts()

# Save the tie points on a shapefile
CRL.tiepoint_grid.to_PointShapefile(path_out='/path/to/store/points/shapefile/file.shp')

# Calculate statistics and save output to txt
with open('STATS_TRIAL_NAME.txt', 'w') as f:
f.write(json.dumps(CRL.tiepoint_grid.calc_overall_stats(include_outliers=False), indent=0))
```

4.3. Hyperspectral / Multispectral Data Fusion

4.3.1. Hyper Pansharpening – Components Substitution Methods

The implementation of Hyper-Sharpening was implemented in python, where multiple functions named GS, GSA_sharpening, broveySharpen, and fihsSharpen were used for all the methods mentioned above. These functions take the two datasets as input and generate fused data with enhanced spatial resolution.

Libraries used to implement the above mentioned Python script include:

- **osgeo**: for handling geospatial data with GDAL
- **numpy**: for numerical operations
- **os**: for operating system functionalities
- **rasterio**: for reading and writing raster data
- **gc**: for garbage collection
- **numpy.matlib**: for matrix manipulation
- **scipy**: for scientific computing
- **scipy.misc**: for miscellaneous functions
- **scipy.signal**: for signal processing (https://docs.scipy.org/doc/scipy/reference/signal.html)
- **PIL (Python Imaging Library):** for image processing (https://pillow.readthedocs.io/en/stable/)
- **cv2 (OpenCV):** for computer vision tasks

4.3.2. Deep Learning for HS-MS Image Fusion

In terms of AFRI4Cast, the resampling of a high resolution signal (Sentinel-2 SRF) to a low resolution signal (Sentinel-2/ PRISMA SR matrix) using full width half maximum (FWHM) values was conducted via the "prospectr" R package. This package is useful for signal processing applications as it offers a diverse range of functions for pre-processing and sample selection of spectral data (https://cran.r-project.org/web/packages/prospectr/prospectr.pdf) .

In particular, the resample2(X, wav, new.wav, fwhm) function was utilized. This function resamples a data matrix or vector to match the response of another instrument using full width half maximum (FWHM) value. Specifically, it resamples the high resolution data to new band positions and resolution using gaussian models defined by fwhm values. It assumes that band spacing and fwhm of the input data is constant over the spectral range. The interpolated values are set to 0 if input data fall outside by 3 standard deviations of the gaussian densities defined by fwhm. The arguments that are used as inputs include:

- **X** : A numeric matrix or vector to resample (optionally a data frame that can be coerced to a numerical matrix).
- **wav** : A numeric vector giving the original band positions.
- **new.wav** : A numeric vector giving the new band positions.
- **fwhm** : A numeric vector giving the full width half maximums of the new band positions. If no value is specified, it is assumed that the fwhm is equal to the sampling interval (i.e. band spacing). If only one value is specified, the fwhm is assumed to be constant over the spectral range.

resample2*(X, wav, new.wav, fwhm)*

Furthermore, a deep neural network was adopted for the implementation of the hyperspectral-multispectral image fusion process. Specifically, AFRI4Cast's HS/MS image fusion was applied by modification of the Pytorch implementation of the following paper: 'Guided Deep Decoder:

Unsupervised Image Pair Fusion' by Tatsumi Uezato, Danfeng Hong, Naoto Yokoya and Wei He, ECCV 2020. The Python code was found in the corresponding GitHub repository, i.e.,

https://github.com/tuezato/guided-deep-decoder/blob/master/

Various Python packages take part in its implementation, which are listed below:

- **matplotlib.pyplot** : A collection of functions that make matplotlib work like MATLAB. pyplot functions for creating a figure or a plotting area within the figure, decorating the plot with labels etc are provided.

- **numpy** : A fundamental package for array computing.

- **pytorch** : A machine and deep learning Python library used for computer vision and natural language processing applications (https://pypi.org/project/torch/) .

- **torch.optim** : A pytorch package for the implementation of various optimization algorithms (https://pytorch.org/docs/stable/optim.html) .

- **scipy.io** : A package that provides essential algorithms for scientific computing in Python. It is designed to work with NumPy arrays and it allows users to have access to a lot of numerical routines, for example routines for numerical integration and optimization (https://docs.scipy.org/doc/scipy/reference/io.html) .

- **skimage.metrics** : A module of scikit-image Python library that in the case of AFRI4Cast's HS/MS image fusion is used to compute the peak signal to noise ratio (PSNR) for an image as well as the mean structural similarity index between two images (https://scikit-image.org/docs/stable/api/skimage.metrics.html) .

- **sklearn.metrics** : A module of scikit Python machine learning library that provides metric functions for the assessment of prediction error. In the case of AFRI4Cast's STTFN, the mean squared error function was used.

5. Inversion of RTMs for Biophysical Parameters

Among the objectives of the Afri4Cast project is the estimation of some biophysical parameters of vegetation, including LAI, from satellite data. These estimates were performed using the PROSAIL software.

For analyzing the biophysical properties of the vegetation with PROSAIL and for the inversion of the RTM, mainly Python packages/libraries were used:

- **progeosail**: this is a fork of the Python bindings of jgomezdans/prosail for the PROSPECT and SAIL leaf and crown reflectance models (https://github.com/XQSDD/progeosail?tab=readme-ov-file) . In addition to the features provided by the original code, the functions of Huemmrich's GeoSail model have been transferred, which utilizes Jasinski's geometric model to represent discontinuous vegetation canopies. Currently, the functions geocone and geocily are available to model cone-shaped and square cylinder-shaped trees respectively, but other shapes can be implemented.

- **GDAL, Numpy, pandas, scikit-learn**

PROSAIL simulations were used to develop a nDecision Tree algorithm to extract LAI information from Sentinel-2 and PRISMA images. In fact, the LAI data retrieved by using the tool available in SNAP has low correspondence with the ground data.

6. Extraction of Data for the Crop Growth and the Rust Prediction Model Through Google Earth Engine (GEE)

Google Earth Engine (GEE) is a web based application in which users can access a big amount of geospatial data. It offers powerful processing capabilities on the available datasets, enabling analysis at a global scale, thus allowing researchers and developers to perform complex tasks without the need for computational resources. Main feature of GEE is the access to a wide range of satellite image collections, from Landsat, Sentinel and MODIS satellites. Covering a wide range of spatial and temporal resolutions these datasets enable users to monitor changes in the Earth's surface over time. GEE provides a toolkit that allows users to perform various tasks, such as spectral index calculations, image classification, detection of changes etc. Being optimized for direct processing on the cloud database of Google, it enables fast processing of large scale datasets. Additionally, it contains other earth observation data, including meteorological information and terrain data, enabling users to combine them for tougher assignments. As a tool, it has been widely adopted in remote sensing applications, performing various tasks, such as deforestation monitoring, crop growth monitoring etc.

In the AFRI4CAst project, GEE was used for the calculation of a wide range spectral indices from Landsat 8 and Sentinel-2 images that were used in the crop growth model and the rust prediction model. Meteorological information derived from the ERA5 reanalysis dataset and soil data from SoilGrids250m 2.0 were also acquired through GEE for the crop growth model, including variables like the minimum and maximum daily temperature, precipitation and reference evapotranspiration. The links for the scripts used during the project are the following:

ERA5-Land Daily Aggregated for the crop growth model:

https://code.earthengine.google.com/55457d62b2ebad4a063c4d49010ba6d5

- Input: parcel coordinates, starting_date, ending_date
- Output: surface_net_solar_radiation_sum (J/day), temperature_2m (K), temperature_2m_max (K), temperature_2m_min (K), total_precipitation_sum (m)

Results were converted at an excel worksheet to:

- Temperature -oC,
- Precipitation- mm,
- Surface net solar radiation - mm/day

SoilGrids 250m 2.0 soil data for the crop growth model:

https://code.earthengine.google.com/be21ba54c665b7292f7148775e8586c2

- Input: point or parcel coordinates
- Output: mean values of sand, silt, clay, soil organic carbon and bulk density at different depths (0-5cm, 5-15cm, 15-30cm, 30-60cm, 60-100cm and 100-200cm) .

Results were extracted from the GEE console and processed at an excel worksheet.

Sentinel-2 spectral indices for the crop growth model:

https://code.earthengine.google.com/c6f887ac0c5013bca4e1c9782342f7b0

- Input: parcel coordinates or parcel_id, starting_date, ending_date
- Output: mean values of NDVI, GreenWDRVI, CC, LAI

Results were extracted from the chart through a .csv file.

Landsat 8 spectral indices for the Rust prediction model:

https://code.earthengine.google.com/3a0cb0fabeb0ac8f68e466bd06660114

- Input: parcel coordinates or parcel_id, starting_date, ending_date

- Output: mean values of AFRI1600, ALTERATION, AVI, ARVI, ARVI2, BWDRVI, CIgreen, CI, DVIMSS, EVI, EVI2, FE2, FE3, GNDVI, GVMI, GARI, GOSAVI, GSAVI, GBNDVI, GRNDVI, IVI, INTENSITY, MVI, normGREEN, normRED, normNIR, PPR, PVR, SIWSI, SLAVI, variGREEN, VEGETATION, WETNESS, WRDVI, WDVI.

Results were extracted from the console and incorporated into an excel worksheet.

END OF THE DOCUMENT