TFMR Retracking Project Code Documentation

Michail-Achillefs Katarachias

2023

Introduction

This document provides the documentation for the functions used in the Python files `file_checking.py`, `function.py`, `main.py` and `libraries`.py .

For this initial version of the project, all the files have to be located in the same directory, as well as the data files to be processed and the **DEM** and `.kml` files (for the LRM mask).

File: file_checking.py

1 File Checking Script Documentation

- **def process_file(file_path)**: This function processes the given file to extract longitude, latitude and power waveform data, and checks if the track of the satellite goes through the defined mask area. It returns a list of valid netCDF file names.
- **kml_file = 'Greenland_LRM_Mask.kml'**: A KML file specifying the mask for the area of Greenland.
- **folder_path = input('Enter the folder path: ')**: Manually input the path of the folder containing the .nc files.
- **with Pool(4) as pool**: Creates a multiprocessing pool that uses 4 CPU cores to process the files in parallel. It can be changed to use as many threads as the user wants.
- **with open('Valid_Names.txt', 'w') as f**: Opens a text file in write mode, and the valid track file names are written to this.

File: main.py

2 Main Script Documentation

- **Importing Libraries and Functions**: Importing the necessary libraries and functions from the `functions.py` module, along with the necessary libraries from `libraries.py`.

- **Reading File Names**: The script reads a list of valid file names for the satellite tracks files from a text file named 'Valid_Names.txt' that was created from the `file_cheking.py` file. The list of file names is stored in a list named `file_list`. Depending on the name of the first file in the list, the mode of operation is determined to be either 'LRM' or 'SAR'.

- **Reading Satellite Data**: The `data_parsing` function is used from the `functions` module to read the necessary parameters like power, time, altitude, window delay, etc., from the first file in `file_list`, which contains satellite measurement data.

- **Reading and Processing DEM Data**: The script imports a Digital Elevation Model (DEM) file named 'arcticdem_mosaic_500m_v3.0.tif', crops the DEM to only select Greenland, and converts it into dictionary format. It also filters the elevation data to remove extreme values.

- **Calculating Range from Satellite to POCA**: For each power waveform measurement, the script calculates the range from the satellite to the Point Of Closest Approach (POCA) using a multiprocessing approach. This is done through a series of functions that find the peak of the power waveform, determine its index, and use this information to calculate the range.

- **Calculating POCA Coordinates**: After obtaining the ranges to POCA for all measurements, the script uses the `NadirToPOCA` function to calculate the corresponding POCA coordinates for each range measurement.

- **Saving Results**: The script saves the coordinates of the nadir (i.e., the point on the ground directly below the satellite) and the POCA for each measurement into separate text files. It also saves a file named 'Output_Data.txt' containing the calculated elevations at each POCA point, the nadir and POCA coordinates, and the corresponding times of the measurements.

- **Visualizing Results**: Finally, the script visualizes the calculated elevations along the track of the satellite over the DEM, and prints out the total execution time.

File: functions.py

3 Function Documentation

3.1 Function: read_kml

This function reads a kml file and returns the coordinates of the polygons in it.

- **Input:** kml_file (str) - The path to the kml file
- **Output:** polygon_coords (list) - The coordinates of the polygons

3.2 Function: is_point_in_mask

This function checks if a point is within a polygon.

- **Input:** polygon_coords (list) - The coordinates of the polygon, point (tuple) - The coordinates of the point
- **Output:** (bool) - True if the point is in the polygon, False otherwise

3.3 Function: data_parsing

This function reads a `.netCDF` file for the track and returns the necessary variables for the implementation of the TFMR.

- **Input:** file (str) - The path to the netCDF file
- **Output:**
  - points (list): The nadir coordinates for each measurement in the track
  - lon (list): The longitude of the nadir points in the track
  - lat (list): The latitude of the nadir points in the track
  - pwr (list): The power waveform of the signal
  - time (list): The time at which the middle of the averaged group of pulses impacts the ground

  - altitude (list): The altitude of the satellite at the time of the measurement
  - window_delay (list): Calibrated 2-way window delay: distance from CoM (Center of Mass) to middle range window

3.4 Function: range_eq

This function calculates the range of the POCA point using the range equation.

Arguments:

- `Tw` (float): The window delay converted to seconds.
- `n` (int): The index of the POCA point in the power waveform.
- `radar_mode` (str): The operating mode of the satellite (LRM or SAR).

Output:

- `R_sat` (float): The range of the POCA point in meters.

3.5 Function: swath_width_LRM

This function calculates the swath width for the LRM radar mode.

Argument:

- `alt` (float): The satellite altitude in meters.

Output:

- Swath area (float): The swath area in square meters.

3.6 Function: filter_elevation_data

This function filters the DEM elevation data to remove outliers.

Arguments:

- `data` (ndarray): The elevation data of the DEM.
- `max_threshold` (int): The maximum elevation threshold in meters.
- `spike_threshold` (int): The maximum elevation difference between two adjacent pixels in meters.

Output:

- `filtered_data` (ndarray): The filtered DEM elevation data.

3.7 Function: crop_dem_Greenland

This function crops the DEM to get only Greenland.

Argument:

- `input_dem` (str): The path to the DEM file.

Output:

- `output_file` (str): The path to the cropped DEM file.

3.8 Function: highest_point_in_swath

(unsused in the script)

This function calculates the highest point in the swath area from the center of the grid.

Arguments:

- `file` (str): The path to the DEM file.
- `point` (tuple): A tuple containing the longitude and latitude of a point.
- `swath_area` (float): The area of the swath.
- `aspect_ratio` (float): The aspect ratio of the swath.

Output:

- `highest_lon` (float): The longitude of the highest point.
- `highest_lat` (float): The latitude of the highest point.
- `dist` (float): The distance from the center of the grid to the highest point.

3.9 Function: DEM_dict

This function reads a DEM file and returns a dictionary with the coordinates of the cells and elevations in the DEM.

Argument:

- `file_path` (str): The path to the DEM file.

Output:

- A dictionary with keys `lon`, `lat`, `z`, representing longitude, latitude, and elevation data of the DEM respectively.

3.10 Function: filter_negative_values

This function filters out negative values from the data.

Arguments:

- `data` (ndarray): The data to be filtered.

Output:

- `filtered_data` (ndarray): The filtered data.

3.11 Function: NadirToPOCA

This function converts the nadir coordinates of a satellite to the POCA coordinates. It also returns the x and y offset coordinates between the nadir and the POCA coordinates.

Arguments:

- `DEM_dict` (dict): The dictionary containing the DEM data.
- `lon` (ndarray): The array of longitude values of the satellite nadir (or the center of the cells in DEM if it is incorporated into the preprocessing algorithms to get the offset grids).
- `lat` (ndarray): The array of latitude values of the satellite nadir (or the center of the cells in DEM if it is incorporated into the preprocessing algorithms to get the offset grids).
- `dxx` (float): The step size in x direction for the DEM interpolation.
- `dyy` (float): The step size in y direction for the DEM interpolation.
- `altitude` (float): The altitude of the satellite.
- `footprint` (float): The footprint radius of the satellite.
- `mt` (str): The method used for the interpolation (griddata function).
- `sm` (float): The smoothing distance used for the Gaussian filter.

Output:

- `Newx` (ndarray): The x-offset coordinates between the nadir and the POCA coordinates.
- `Newy` (ndarray): The y-offset coordinates between the nadir and the POCA coordinates.

- `Newlon` (ndarray): The longitude of the POCA coordinates.
- `Newlat` (ndarray): The latitude of the POCA coordinates.

File: libraries.py

4 Importing of libraries

- **sys, os**: Standard Python libraries for interacting with the Python interpreter and the operating system respectively. They provide functions for file and directory manipulation, environment variable access, and command-line argument parsing among others.
- **functions**: A custom Python module containing specific functions for this project.
- **numpy**: A library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- **matplotlib.pyplot**: A module in the matplotlib library for creating static, animated, and interactive visualizations in Python.
- **geopy.point, geopy.distance**: Modules from the Geopy library used for working with points, distances, and calculations related to geographical points.
- **lxml.etree**: This is a Pythonic binding for the C libraries libxml2 and libxslt. It provides safe and convenient access to these libraries using the ElementTree API.
- **shapely.geometry**: This module from the Shapely library provides classes for the manipulation and analysis of planar geometric objects.
- **netCDF4**: A Python interface to the netCDF library, used for reading and writing files in the netCDF format, a data format for storing array-oriented scientific data.
- **math, scipy**: These are mathematical libraries in Python that provide functions for numerical calculations, such as interpolation, integration, optimization, image processing, statistics, and more.

- **multiprocessing, concurrent.futures**: These libraries provide support for parallel processing in Python, allowing the program to utilize multiple cores of the computer's CPU.
- **tqdm**: A Python library for displaying progress bars in the console.
- **pyproj, rasterio, osgeo**: These libraries provide Python interfaces for manipulating geospatial data.
- **numba**: A Just-In-Time compiler for Python that translates a subset of Python and NumPy code into fast machine code.
- **datetime**: A Python module for manipulating dates and times.
- **scipy.ndimage**: A submodule of SciPy that contains various functions for multi-dimensional image processing.