OBIA4RTM Documentation

Release 1.0.0

Lukas Graf

Apr 23, 2023

CONTENTS:

Created on Sat Jul 13 09:44:14 2019

This module is part of OBIA4RTM.

@author: Lukas Graf, graflukas@web.de

`OBIA4RTM.configurations.connect_db.`**`close_db_connection`**`(`*con*, *cursor*`)`

closes an opened database connection

Parameters

con
[psycopg2 Database Connection] connection to be closed

cursor
[psycopg2 Database Cursor] cursor to be closed

`OBIA4RTM.configurations.connect_db.`**`connect_db`**`()`

connect to PostgreSQL database by using the specifications in 'postgres.ini' File in the root of the OBIA4RTM home directory stored in the user-profile

Returns

conn
[psycopg2 Database connection] connection object to PostgreSQL database

cursor psycopg2 Database cursor
cursor for querying and inserting data from and to PostgreSQL DB

`OBIA4RTM.configurations.connect_db.`**`get_db_connection_details`**`()`

reads and returns the postgres.ini connection details

CHAPTER

ONE

RETURNS:

parser
[ConfigParser Object] parsed database configurations from postgres.ini file

Created on Tue Jul 30 15:38:37 2019

This module is part of OBIA4RTM.

Copyright (c) 2019 Lukas Graf

@author: Lukas Graf, graflukas@web.de

`OBIA4RTM.configurations.db_management.`**`update_luc_table`**(*landcover_table*, *landcover_cfg=None*)

updates the land-cover/ land use table in OBIA4RTM that is required for performing land-cover class specific vegetation parameter retrieval Make sure that the classes in the config file match the land cover classes provided for the image objects and used for generating the lookup-table. Otherwise bad things might happen.

NOTE: in case land cover classes that are about to be inserted are already stored in the table, they will be overwritten!

Parameters

landcover_table
[String] name of the table with the land cover information (<schema.table>)

landcover_cfg
[String] file-path to land cover configurations file

Returns

None

Created on Fri Jul 5 14:11:59 2019

This module is part of OBIA4RTM.

Copyright (c) 2019 Lukas Graf

@author: Lukas Graf, graflukas@web.de

`OBIA4RTM.configurations.logger.`**`close_logger`**(*logger*)

close a logger after program shut-down and releases the handlers

Parameters

logger
[logging Object] logger of OBIA4RTM 'OBIA4RTM_logger'

OBIA4RTM.configurations.logger.**determine_logdir**()

searches the logging directory used for OBIA4RTM

Returns

log_dir
[String] Path of logging directory

OBIA4RTM.configurations.logger.**get_logger**(*logname=None*)

setups up a new logging object using Rotating File Handlers

Parameters

OBIA4RTM_log_dir
[String] directory, the log-file should be written to

logname
[String] name of the logger (opt.); per default OBIA4RTM_Logger will be used

Returns

logger
[logging Logger] Logger with stream handler for tracing OBIA4RTM's activities and errors

Created on Sat Mar 9 09:03:14 2019

This module is part of OBIA4RTM.

@author: Lukas Graf, graflukas@web.de

OBIA4RTM.inversion.distributions.**gaussian**(*minimum*, *maximum*, *num*, *mean*, *std*)

draws a truncated gaussian distribution between min and max

Parameters

minimum
[float] lower bound of the truncated Gaussian distribution

maximum
[float] upper bound of the truncated Gaussian distribution

num
[Integer] number of samples to be drawn

mean
[float] centre of the truncated Gaussian distribution

std
[flaot] standard deviation, controlls the width of the distribution betweem min and max

Returns

truncated
[np.array] Array with values drawn from the truncated Gaussian distribution

OBIA4RTM.inversion.distributions.**uniform**(*minimum*, *maximum*, *num*)

draws a uniform distribution between min and max

Parameters

minimum
[float] lower bound of the uniform distribution

maximum
[float] upper bound of the uniform distribution

num
[Integer] number of samples to be drawn

Returns

uni
[np.array] Array with values drawn from the uniform distribution

Created on Sat Mar 9 10:58:16 2019

This module is part of OBIA4RTM.

`OBIA4RTM.inversion.handle_metadata.`**`get_bands`**`(`*conn*, *cursor*, *sensor*, *logger*`)`

reads in sensor band centers and FWHM stored in database

Parameters

conn
[psycopg2 Database connection] connection to OBIA4RTM PostgreSQL database

cursor
[psycopg2 Database cursor] cursor for DB inserts and queries

sensor
[String] name of the sensor; currently either 'S2A' or 'S2B'

logger
[logging Logger] for recording errors to the log file

Returns

centers
[List] list of central wavelengths of the spectral bands of the sensor (nm)

fwhm
[List] list of the full width half maximum of the spectral bands (nm)

`OBIA4RTM.inversion.handle_metadata.`**`get_resampler`**`(`*conn*, *cursor*, *sensor*, *logger*`)`

get the spectral properties of a sensor and generate a resampler object. Currently, Sentinel-2A and Sentinel-2B are supported

Parameters

conn
[psycopg2 Database connection] connection to OBIA4RTM PostgreSQL database

cursor
[psycopg2 Database cursor] cursor for DB inserts and queries

sensor
[String] name of the sensor; currently either 'S2A' or 'S2B'

logger
[logging Logger] for recording errors to the log file

Returns

resampler
[spectral BandResampler] Resampler Object for resampling the ProSAIL output to the spectral resolution of Sentinel-2

Created on Sat Jul 13 11:14:32 2019

This module is part of OBIA4RTM.

@author: Lukas Graf, graflukas@web.de

OBIA4RTM.inversion.handle_prosail_cfg.**get_landcover_classes**(*landcover_cfg*)

get the number and names of the land cover classes stored in the ProSAIL.cfg file

Parameters

cfg_file
[String] path to the land cover configuration file

Returns

luc_classes
[List] list of landcover classes (code + semantics)

OBIA4RTM.inversion.handle_prosail_cfg.**read_params_per_class**(*prosail_cfg*, *landcover_cfg*, *logger*)

reads in the vegetation parameters for the ProSAIL model for each land cover class

Parameters

cfg_file
[String] path to the ProSAIL configurations file

landcover_cfg
[String] path to the landcover configuration file

logger
[logging.Logger] for recording errors

Returns

container
[Dictionary] dict with the ProSAIL parameters per land cover class

Created on Sat Mar 9 10:34:57 2019

This module is part of OBIA4RTM.

@author: Lukas Graf, graflukas@web.de

class OBIA4RTM.inversion.inversion.**inversion**(*scene_id*)

super-class for the object-based inversion of satellite scenes

Methods

do_inversion(land_use, num_solutions, ...[, ...])	performs inversion on all objects for a given date.
do_obj_inversion(object_id, acqui_date, ...)	performs inversion per single object using mean of xx best solutions (RMSE criterion) and stores result results table params to be inverted/ returned should be passed as list of strings e.g: inv_params = ["LAI", "CAB"] also inverted spectra can be returned: therefore just append the band numbers to the list of strings of parameters: e.g.
gen_lut(inv_mapping_table, inv_table[, ...])	Generates the lookup table and stores it in the DB must be run seperately from the inversion part
get_scene_metadata()	queries the relevant scene metadata from the database
set_ProSAIL_config([path_to_config])	read in the config file holding the vegetation parameters for setting up the lookup table using the ProSAIL radiative transfer model
set_landcover_config([path_to_lc_config])	read in the land cover config file holding the land cover classes for setting up the lookup table using the ProSAIL radiative transfer model
set_soilrefl([path_to_soilrefl_file])	set up the file-path to the txt file containing the soil-reflectance required for ProSAIL to account for the soil background and read in the values

do_inversion(*land_use*, *num_solutions*, *res_table*, *object_table*, *inv_mapping_table*, *lut_table*, *return_specs=True*)

performs inversion on all objects for a given date. NOTE: the object reflectance values must be already available in the data base. Run gen_lut therefore before! Works as a wrapper around the do_object_inversion method

Parameters

lande_use
[Integer] land cover code for the specific object and date

num_solutions
[Integer] how many solutions should be used for generating the inversion result

res_table
[String] tablename where to store the results of the inversion (<schema.table>)

object_table
[String] tablename of table containing the object spectra (<schema.table>)

inv_mapping_table
[String] tablename of the table containing the parameters to be inverted per acqusition date (scene) and land use/ cover class

lut_table
[String] table containing the ProSAIL lut on a per scene and landuse / cover class base

return_specs
[Boolean] determines whether inverted spectra should be returned (True; default)

Returns

None

`do_obj_inversion`(*object_id*, *acqui_date*, *land_use*, *num_solutions*, *inv_params*, *res_table*, *object_table*, *lut_table*)

performs inversion per single object using mean of xx best solutions (RMSE criterion) and stores result results table params to be inverted/ returned should be passed as list of strings e.g: inv_params = ["LAI", "CAB"] also inverted spectra can be returned: therefore just append the band numbers to the list of strings of parameters: e.g. inv_params = ["LAI", "CAB", "B2", "B3", etc.]

Parameters

object_id
[Integer] ID of the current object (derived from OBIA4RTM database)

acqui_date
[Date (YYYY-MM-DD)] acquisition date of the image used for the inversion

lande_use
[Integer] land cover code for the specific object and date

num_solutions
[Integer] how many solutions should be used for generating the inversion result

inv_params
[List] list of the parameters (must be named) to be inverted

res_table
[String] tablename where to store the results of the inversion (<schema.table>)

object_table
[String] tablename of the table containing the object spectra (<schema.table>)

lut_table
[String] tablename of the lookup-table (<schema.table>)

Returns

status
[Integer] zero if everything is OK

`gen_lut`(*inv_mapping_table*, *inv_table*, *landcover_config_path=None*, *prosail_config_path=None*, *soil_path=None*)

Generates the lookup table and stores it in the DB must be run seperately from the inversion part

Parameters

inv_mapping_table
[String] name of the table storing the inversion mapping required for performing the inversion

inv_table
[String] Name of the table the lookup-table should be written to (<schema.table>)

landcover_config_path
[String] file-path to landcover config file (opt.; per default the OBIA4RTM delivered file will be used)

prosail_config_path
[String] file-path to landcover config file (opt.; per default the OBIA4RTM delivered file will be used)

soil_path
[String] file-path to file with soil reflectance values (opt.; per default the OBIA4RTM delivered file will be used)

`get_scene_metadata()`

queries the relevant scene metadata from the database

`set_ProSAIL_config(path_to_config=None)`

read in the config file holding the vegetation parameters for setting up the lookup table using the ProSAIL radiative transfer model

Parameters

path_to_config
[String] optinal, path and filename of config-file for ProSAIL

Returns

path_to_config
[String] definite location of the config file or error if file not found

`set_landcover_config(path_to_lc_config=None)`

read in the land cover config file holding the land cover classes for setting up the lookup table using the ProSAIL radiative transfer model

Parameters

path_to_lc_config
[String] optinal, path and filename of config-file for land cover classes

Returns

path_to_config
[String] definite location of the config file or error if file not found

`set_soilrefl(path_to_soilrefl_file=None)`

set up the file-path to the txt file containing the soil-reflectance required for ProSAIL to account for the soil background and read in the values

Parameters

path_to_soilrefl_file
[String] optional, file to the txt file with soil reflectance values

Returns

soils
[Numpy Array] array of soil reflectance values (1 nm steps)

Created on Sat Mar 9 06:52:00 2019

This module is part of OBIA4RTM.

@author: Lukas Graf, graflukas@web.de

`class OBIA4RTM.inversion.lookup_table.lookup_table`

class for creating and storing biophysical parameters in a lookup table (LUT) like structure

Methods

generate_param_lut(params)	get the minima, maxima, number and distribution type of parameters to be inverted and prepares them for storing in a LUT accordingly

generate_param_lut(*params*)

get the minima, maxima, number and distribution type of parameters to be inverted and prepares them for storing in a LUT accordingly

Parameters

params
[numpy array] Array containing the ProSAIL parameters extracted from cfg file

Created on Fri Jul 26 15:42:10 2019

This module is part of OBIA4RTM.

Copyright (c) 2019 Lukas Graf

@author: Lukas Graf, graflukas@web.de

OBIA4RTM.S2_PreProcessor.install_6S.**install_6S**(*link=None*)

installs the 6S algorithm required for atmospheric correction from the Fortran source code using gfortran. The sources will be installed to the OBIA4RTM install dir (somewhere in the user profile) where also the config files are stored.

Currently, only Linux/ Unix systems are supported for this operation.

In case you expire any problems automatically downloading and building 6S, please also consult: https://py6s.readthedocs.io/en/latest/installation.html

Windows-Users please note that the installation might not be that smoothly as on Posix, as you will have to install a bunch of additional functionalities

Created on Wed Jul 31 12:32:24 2019

This module is part of OBIA4RTM. It is part of the (optional) Sen2Core preprocessing wrapper addon that takes care about image preprocessing (i.e atmospheric correction) using Sen2Core software (see: http://step.esa.int/main/third-party-plugins-2/sen2cor/)

NOTE: Sen2Core must be installed on your computer!

NOTE: Windows users might have to add the GDAL binaries (they come with OSGEO) to their PATH variable

Copyright (c) 2019 Lukas Graf

@author: Lukas Graf, graflukas@web.de

OBIA4RTM.S2_PreProcessor.s2_sen2core_attcor.**call_gdal_merge**(*sentinel_data_dir_l2*, *resolution*, *storage_dir=None*)

calls the gdal_merge.py script to make an image layer stack and prepare the imagery for usage in OBIA4RTM. Outputs a GeoTiff with the nine Sentinel-2 bands used in OBIA4RTM and the SCL band that contains the pre-classification information.

You can use this function also if you L2 data

Parameters

sentinel_data_dir_l2
[String] path of the directory containing the output of sen2core in L2 level

resolution
[Integer] spatial resolution of the atmospherically corrected imagery possible value: 10, 20, 60 meters

storage_dir
[String] path to the directory the layer stack should be moved to. If None, the layer stack will remain the sentinel_data_dir_l2 in the img folder

Returns

fname_stack
[String] file-path to the stacked imagery

metadata_xml
[String] file-path to the metadata xml file

OBIA4RTM.S2_PreProcessor.s2_sen2core_attcor.**call_sen2core**(*sentinel_data_dir*, *zipped*, *resolution*, *path_sen2core*)

calls Sen2Core and runs it on a downloaded Sentinel 1C dataset to convert it to Level 2A. The output spatial resolution must be provided (10, 20 or 60 meters)

NOTE: If you have already L2 imagery then only run the gdal_merge wrapper and to not use this function

Parameters

sentinel_data_dir
[String] path to the directory that contains the Level-1C data. In case the data is zipped (default when downloaded from Copernicus) specify the file-path of the zip

zipped
[Boolean] specifies if the directory with the Sat data is zipped

resolution
[Integer] spatial resolution of the atmospherically corrected imagery possible value: 10, 20, 60 meters

path_sen2core
[String] directory containing Sen2Core software (top-most level; e.g. /home/user/Sen2Core/). Must be the same directory as specified during the Sen2Core installation process using the –target option

Returns

sentinel_data_dir_l2

OBIA4RTM.S2_PreProcessor.s2_sen2core_attcor.**check_sen2core_installation**(*path*)

check if Sen2Core is installed and can be used

Parameters

path
[String] user-provided path to Sen2Core installation directory

Returns
———-
exists
[Boolean] True, if directory exists and False if tests fail

cmd
[String] command for testing if Sen2Core is working

OBIA4RTM.S2_PreProcessor.s2_sen2core_attcor.**do_sen2core_preprocessing**(*sentinel_data_dir*, *zipped*, *resolution*, *path_sen2core*, *storage_dir=None*)

calls the sen2core wrapper and the gdal_merge wrapper to carry out the atmospheric correction and preclassification on Sentinel-1 imagery (L1C level) required for OBIA4RTM.

NOTE: If you have already L2 imagery then only run the gdal_merge wrapper and to not use this function Make also sure that gdal_merge can be executed from the command line.

NOTE: Windows users might have add the gdal binaries to their PATH!!

Parameters

sentinel_data_dir
[String] path to the directory that contains the Level-1C data. In case the data is zipped (default when downloaded from Copernicus) specify the file-path of the zip

zipped
[Boolean] specifies if the directory with the Sat data is zipped

resolution
[Integer] spatial resolution of the atmospherically corrected imagery possible value: 10, 20, 60 meters

path_sen2core
[String] directory containing Sen2Core software (top-most level; e.g. /home/user/Sen2Core/). Must be the same directory as specified during the Sen2Core installation process using the –target option

storage_dir
[String] path to the directory the final layer stack should be moved to. If None, the layer stack will remain the sentinel_data_dir_l2 in the img folder

Returns

fname_stack
[String] file-path to the stacked imagery (required in OBIA4RTM)

metadata_xml
[String] file-path to the metadata xml file (required in OBIA4RTM)

Created on Thu Jul 25 10:15:38 2019

This module is part of OBIA4RTM.

@author: Lukas Graf, graflukas@web.de

OBIA4RTM.setup_db.create_schema.**check_if_exists**(*schema*, *table_name*, *cursor*)

auxiiliary function to check whether a given table exists in a given schema

Parameters

schema
[String] name of the schema the table should be created in

table_name
[String] name of the table to be created

cursor
[psycopg2 Database Cursor] for querying the database

Returns

exists
[Boolean] True, if table already exists, False else

OBIA4RTM.setup_db.create_schema.**create_function_statement**(*sql_function*, *logger*)

create a SQL statement for creating/ replacing a SQL function

Parameters

sql_function
[String] file-path to the sql-function

logger
[logging.Logger] for logging errors

Returns

sql_statement
[String] processed and ready-to-execute sql statement

OBIA4RTM.setup_db.create_schema.**create_schema**()

this function is used to generate a new schema in the OBIA4RTM database. In case the schema already exists, nothing will happen. The schema to be created is taken from the obia4rtm_backend.cfg file

Parameters

None

Returns

status
[integer] zero if everything was OK

OBIA4RTM.setup_db.create_schema.**create_sql_statement**(*sql_file*, *schema*, *table_name*, *logger*)

auxiiliary function to create the sql_statement required to create the specific tables in the DB schema

Parameters

sql_file
[String] file-path to the sql-template containing the sql-statement for creating the table

schema
[String] name of the schema the table should be created in

table_name
[String] name of the table to be created

logger
[logging.Logger] for logging errors

Returns

sql_statement
[String] processed and ready-to-execute sql statement

Created on Thu Jul 18 15:08:29 2019

This module is part of OBIA4RTM.

@author: Lukas Graf, graflukas@web.de

class OBIA4RTM.setup_db.setup_postgres.**setupDataBase**

class for setting up a PostgresSQL database as backend for OBIA4RTM

Methods

connect_to_postgres()	connects to default Postgres database running on specified host in postgres.ini file to create the OBIA4RTM Postgres database
create_OBIA4RTM_DB()	create the OBIA4RTM database using the specification of the postgres.uni file
enable_extensions()	enables PostGIS and HSTORE extension required for OBIA4RTM
setup_backend()	runs the whole setup-procedure for creating the OBIA4RTM backend
setup_public_functions()	setups the RMSE function used in OBIA4RTM for doing the inversion
setup_public_tables()	setups all those tables, that are required in the public schema of the OBIa4RTM backend database

connect_to_postgres()

connects to default Postgres database running on specified host in postgres.ini file to create the OBIA4RTM Postgres database

Returns

con
[psycopg2 Database Connection] Connection to DEFAULT Postges database (not OBIA4RTM database)

cursor
[psycopg2 Database Cursor] Cursor for this default database

create_OBIA4RTM_DB()

create the OBIA4RTM database using the specification of the postgres.uni file

Returns

status
[Integer] zero, if everything was OK

enable_extensions()

enables PostGIS and HSTORE extension required for OBIA4RTM

setup_backend()

runs the whole setup-procedure for creating the OBIA4RTM backend

setup_public_functions()

setups the RMSE function used in OBIA4RTM for doing the inversion

setup_public_tables()

setups all those tables, that are required in the public schema of the OBIa4RTM backend database

Created on Fri Jul 19 10:43:55 2019

This module is part of OBIA4RTM.

@author: Lukas Graf, graflukas@web.de

OBIA4RTM.install.**install**(*install_addons=False*)

does a full-installation of OBIA4RTM backend facilities including the PostgreSQL database setup and copying of configuration files to a user- accessible directory NOTE: PostgreSQL must be almost installed as well as PostGIS

Parameters

install_addons

[Boolean] Def: False; if True 6S for atmospheric correction together with GEE will be installed -> please read the requirements before!

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

O

INDEX

M

O

R

S

U