SenseTrack

Indice

Offset-Tracking Submodule (ot)

Overview

Il sottopacchetto `sensetrack.ot` fornisce le funzionalità principali per l'analisi di optical flow, la normalizzazione delle immagini, la gestione delle interfacce e la CLI per l'offset tracking. È progettato per lavorare con immagini satellitari e dati raster, offrendo algoritmi avanzati e strumenti di supporto per la ricerca e l'applicazione operativa.

Struttura del modulo

- `algoritmi.py`
  Implementa le classi e le funzioni per il calcolo dell'optical flow e della cross-correlazione di fase tra immagini. Fornisce wrapper per algoritmi OpenCV e scikit-image, oltre a utilità per la conversione dei risultati in DataFrame o GeoDataFrame.
- `cli.py`
  Implementa la Command Line Interface per lanciare i processi di optical flow, normalizzazione e altre operazioni direttamente da terminale.
- `helpmsg.py`
  Contiene i messaggi di aiuto e la documentazione testuale per la CLI e le funzioni principali del modulo.
- `interfaces.py`
  Definisce le classi per la rappresentazione delle immagini, la gestione delle bande e le interfacce astratte per gli algoritmi di tracking.
- `lib.py`
  Funzioni di supporto e utilità comuni per la manipolazione di immagini, conversioni di formato e operazioni matematiche ricorrenti.
- `metodi.py`
  Factory per la creazione di istanze degli algoritmi di optical flow. Permette di selezionare e configurare dinamicamente l'algoritmo desiderato tramite nome o parametri.
- `opencvof.py`
  Implementazione degli algoritmi di optical flow basati su OpenCV (es.

Farneback, Lucas-Kanade). Consente la configurazione dettagliata dei parametri e l'integrazione con pipeline di elaborazione.

- `skiofilk.py`
  Implementazione dell'algoritmo ILK (Iterative Lucas-Kanade) tramite scikit-image.
- `skioftvl1.py`
  Implementazione dell'algoritmo TV-L1 (Total Variation L1) tramite scikit-image, robusto per immagini rumorose e variazioni di intensità.
- `skipccv.py`
  Implementazione della Phase Cross-Correlation (PCC Vector) per la stima di spostamenti sub-pixel tra immagini. A differenza degli altri algoritmi che restituiscono mappe di spostamento in formato raster, `skipccv` restituirà un file georiferito vettoriale in cui ogni punto rappresenta il centro della finestra di ricerca, e sarà associato agli spostamenti nelle due direzioni principali (campi `RSHIFT` e `CSHIFT`) lo spostamento risultante (campo `L2`) e lo scarto quadratico medio normalizzato tra le finestre mobili analizzate (campo `NRMS`).
- `image_processing/`
  Sotto-pacchetto con moduli per la normalizzazione, l'equalizzazione, la conversione e la manipolazione avanzata di immagini raster e array.

Funzionalità principali

- Calcolo di optical flow tra immagini raster (OpenCV, scikit-image)
- Normalizzazione e trasformazione di bande e immagini
- Equalizzazione e conversione di immagini per l'analisi
- Interfacce astratte per la gestione di immagini multi-banda
- CLI per lanciare processi di tracking e normalizzazione
- Supporto per la conversione dei risultati in formati tabulari (DataFrame, GeoDataFrame)

Esempio di utilizzo

```
from sensetrack.ot.metodi import get_algorithm
from sensetrack.ot.interfaces import Image
from sensetrack.lib import image_to_rasterio, rasterio_open, basic_pixel_coregistration
from sensetrack.ot.algoritmi import OpenCVOpticalFlow

# Allineamento dei pixel dell'immagine target rispetto all'immagine reference
# Scrive il file coregistrato (`tar_coreg.tif`)
basic_pixel_coregistration("ref.tif", "tar.tif", "tar_coreg.tif")
ref = Image(**rasterio_open("ref.tif", band = 1), nodata = -9999.)
tar = Image(**rasterio_open("tar_coreg.tif", band = 1), nodata = -9999.)
OT = OpenCVOpticalFlow.from_dict({"pyr_scale":0.5, "levels":4, "winsize":16})
result = OT(reference=ref.get_band("B1"), target=tar.get_band("B1"))
image_to_rasterio(result, "output.tif")
```

CLI

Per lanciare la medesima analisi da terminale (la coregistrazione viene comunque eseguita):

```
python -m sensetrack.ot.cli --algname OPENCVOF
  --reference ref.tif --target tar.tif
  --output output.tif --winsize 16 --levels 4
```

Algoritmi implementati

OpenCV Optical Flow (Farneback, Lucas-Kanade)

- Implementato in `opencvof.py`.
- Algoritmi classici per optical flow densi e sparsi.
- Parametri configurabili: `pyr_scale`, `levels`, `winsize`, `iterations`, `poly_n`, `poly_sigma`, `flags`.
- Adatto a immagini con variazioni di intensità moderate e movimenti continui.

Scikit-Image ILK (Iterative Lucas-Kanade)

- Implementato in `skiofilk.py`.
- Algoritmo iterativo per optical flow, robusto su immagini scientifiche e remote sensing.
- Parametri: `radius`, `num_warp`, `gaussian`, `prefilter`.
- Indicato per piccoli spostamenti e immagini con basso rumore.

Scikit-Image TV-L1 (Total Variation L1)

- Implementato in `skioftvl1.py`.
- Algoritmo robusto per optical flow su immagini rumorose o con variazioni di intensità.
- Parametri: `attachment`, `tightness`, `num_warp`, `num_iter`, `tol`, `prefilter`.
- Ideale per dati SAR e immagini con discontinuità.

Phase Cross-Correlation (PCC)

- Implementato in `skipccv.py`.
- Algoritmo per la stima di spostamenti sub-pixel tramite cross-correlazione di fase.
- Parametri: `winsize`, `step_size`, `phase_norm`, `upsmp_fac`.
- Utile per misurazioni di offset precisi tra immagini allineate.

Note

- Tutti gli algoritmi sono configurabili tramite parametri Python o CLI.

- Il modulo è pensato per essere estendibile: è possibile aggiungere nuovi algoritmi implementando le interfacce previste.
- L'output può essere salvato in diversi formati, inclusi raster e DataFrame.

SNAP-GPT Submodule (snap_gpt)

Overview

Il sottopacchetto `sensetrack.snap_gpt` fornisce strumenti e workflow per il preprocessamento di dati SAR (Synthetic Aperture Radar) tramite l'integrazione con SNAP-GPT (Graph Processing Tool) di ESA. Permette la gestione automatizzata di grafi di elaborazione, la definizione di aree di interesse (AOI), la configurazione di parametri e l'esecuzione batch di processi SAR.

Struttura del modulo

- `lib.py`
  - Funzioni e classi per la gestione dei workflow SNAP-GPT, inclusa la creazione, modifica ed esecuzione di grafi di elaborazione.
  - Gestione delle chiamate a SNAP-GPT da Python, parsing dei log e gestione degli errori.
  - Utility per la generazione di comandi, la gestione dei file temporanei e la verifica dei risultati.

Funzionalità principali

- **Gestione workflow SNAP-GPT**: automatizza la creazione e l'esecuzione di grafi SNAP per preprocessamento SAR (ad es. Sentinel-1, COSMO-SkyMed).
- **Configurazione flessibile**: parametri, percorsi e AOI sono gestiti tramite file YAML e GeoPackage.
- **Supporto multi-sensore**: workflow predefiniti per diversi sensori (S1, COSMO, S2) tramite file XML di grafo.
- **Batch processing**: esecuzione di processi su più file SAR in modo automatico e ripetibile.
- **Gestione AOI**: selezione e applicazione di aree di interesse per il subset delle immagini.
- **Logging e gestione errori**: parsing dei log SNAP-GPT, gestione delle eccezioni e reportistica.

Esempio di utilizzo

```
import geopandas as gpd
from sensetrack.snap_gpt.lib import SARPreprocessing, AOI

# Seleziona area di interesse e processo
```

```
subset = gpd.read_file("./aoi.shp")
process = SARPreprocessing.S1_SLC_DEFAULT
sarfile = "path/to/sarfile.zip"

# Crea e lancia il preprocessamento
preproc = SARPreprocessing(subset, process)
preproc.run(sarfile)
```

Descrizione dei grafi XML SNAP-GPT

Di seguito una sintesi dei principali nodi presenti nei workflow XML forniti con il modulo:

`s1_slc_default.xml`

- **Read**: Lettura del file di input SLC.
- **ApplyOrbit**: Applicazione dell'orbita precisa.
- **ThermalNoiseRemoval**: Rimozione del rumore termico.
- **Calibration**: Calibrazione radiometrica.
- **TOPSAR-Deburst**: Ricostruzione delle immagini burst.
- **Multilook**: Riduzione della risoluzione spaziale tramite multilook.
- **Speckle-Filter**: Filtro per la riduzione del rumore speckle.
- **TerrainCorrection**: Ortorettifica e correzione topografica.
- **Subset**: Estrazione di una regione di interesse.
- **Write**: Scrittura del risultato su file GeoTIFF.

`s1_slc_default_noSF.xml`

- Come sopra, ma senza filtro speckle.

`s1_slc_default+b3.xml`

- Come `s1_slc_default.xml` fino a TerrainCorrection.
- **BandMaths**: Calcolo di una banda aggiuntiva (es. rapporto tra polarizzazioni).
- **BandMerge**: Fusione delle bande originali e calcolate.
- **Write**: Output finale.

`s1_slc_default+b3noSF.xml`

- Come `s1_slc_default+b3.xml` ma senza filtro speckle.

`s1_grd_default.xml`

- **Read**: Lettura prodotto GRD.
- **ApplyOrbit**: Applicazione orbita.
- **ThermalNoiseRemoval**: Rimozione rumore termico.
- **TerrainCorrection**: Ortorettifica.

- **Write**: Output finale.

`s2_l2a_default.xml`

- **Read**: Lettura prodotto Sentinel-2.
- **Resample**: Ricampionamento a 10m.
- **BandSelect**: Selezione delle bande principali.
- **Subset**: Estrazione AOI.
- **Write**: Output finale.

`cosmo_scs-b_default.xml`

- **Read**: Lettura prodotto Cosmo-SkyMed.
- **LinearToFromdB**: Conversione tra scala lineare e dB.
- **Multilook**: Riduzione risoluzione.
- **Terrain-Correction**: Ortorettifica.
- **Subset**: Estrazione AOI.
- **Write**: Output finale.

Note

- I grafi SNAP (file XML) sono personalizzabili e possono essere estesi per nuovi workflow.
- Il modulo è pensato per essere integrato in pipeline più ampie di elaborazione dati SAR.
- Per l'uso di SNAP-GPT è necessario che SNAP sia installato e accessibile dal sistema.

Sentinel-1 SAR Preprocessing Submodule (sentinel)

Overview

Il sottopacchetto `sensetrack.sentinel` fornisce strumenti e classi per il preprocessamento di dati SAR provenienti dal satellite Sentinel-1. Consente la gestione automatizzata di workflow di elaborazione, la lettura e manipolazione dei manifest, e l'integrazione con pipeline di analisi più ampie.

Struttura del modulo

- `preprocessing.py`
  - Classe principale: `S1Preprocessor`
  - Gestisce l'intero workflow di preprocessamento per dati Sentinel-1 SLC e GRD.
  - Permette la configurazione di parametri, la selezione di aree di interesse (AOI) e l'integrazione con SNAP-GPT.
  - Supporta l'esecuzione batch su più file SAR.

- `manifest_file.py`
  - Funzioni e classi per la lettura, il parsing e la manipolazione dei file manifest XML di Sentinel-1.
  - Estrae metadati, parametri di acquisizione, geometrie e informazioni di orbita utili per il preprocessamento e l'analisi.

Funzionalità principali

- **Preprocessamento SAR Sentinel-1**: automatizza la catena di elaborazione (calibrazione, ortorettifica, filtraggio, subset AOI, ecc.)
- **Gestione manifest**: parsing avanzato dei file manifest XML per estrarre metadati e parametri di scena.
- **Configurazione flessibile**: parametri e workflow personalizzabili tramite file di configurazione e classi Python.
- **Integrazione con SNAP-GPT**: utilizzo di grafi XML per lanciare processi SNAP direttamente da Python.
- **Supporto batch**: elaborazione di più scene in sequenza automatica.

Esempio di utilizzo

```
import geopandas as gpd
from sensetrack.s1.preprocessing import S1Preprocessor
# Carica un ESRI shapefile poligonale per la definizione dell'area di interesse (AOI)
aoi = gpd.read_file("./aoi.shp")
# Crea un preprocessor per una specifica AOI e processo
preproc = S1Preprocessor(aoi=aoi, process="S1_SLC_DEFAULT")
preproc.run("path/to/sarfile.zip")
```

Note

- Il modulo richiede che SNAP sia installato e accessibile dal sistema.
- I workflow possono essere estesi per includere ulteriori step di elaborazione (ad es. speckle filtering, coregistrazione, ecc.).
- Il parsing dei manifest consente di automatizzare la selezione di parametri e la verifica di qualità dei dati.

COSMO-SkyMed Submodule (cosmo)

Overview

Il sottopacchetto `sensetrack.cosmo` fornisce strumenti avanzati per la gestione, il preprocessamento e l'analisi di dati SAR provenienti dai satelliti COSMO-SkyMed di prima (CSK) e seconda generazione (CSG). Include classi per la decodifica dei metadati, la manipolazione dei file HDF5, l'estrazione di footprint, la conversione di quicklook e l'integrazione con workflow SNAP-GPT.

Struttura del modulo

- `lib.py`:
  - Definisce le classi principali per la rappresentazione e il parsing dei prodotti COSMO-SkyMed (CSK/CSG), incluse le enumerazioni per polarizzazione, orbita, tipo prodotto e squint.
  - Classi:
    - `CSKProduct` e `CSGProduct`: parsing avanzato dei nomi file, estrazione metadati, gestione delle convenzioni di denominazione.
    - `CSKFile`: estende `h5py.File` per fornire accesso diretto a quicklook, bande, footprint e metadati.
    - `Pols`: dizionario specializzato per la gestione delle polarizzazioni.
- `utils.py`:
  - Funzioni di utilità per:
    - Parsing e manipolazione di file HDF5 (lettura attributi, esplorazione struttura, estrazione shape e footprint).
    - Conversione di quicklook in immagini, esportazione batch, creazione di GeoDataFrame dei footprint.
    - Calcolo dell'angolo di incidenza medio.
- `preprocessing.py`:
  - Classi `CSKPreprocessor` e `CSGPreprocessor`: gestiscono il preprocessamento di prodotti CSK/CSG tramite workflow SNAP-GPT, con stima automatica dei parametri multilook, selezione AOI, proiezione custom (CRS) e generazione di output georiferiti.
  - CLI unificata per processare file HDF5 CSK/CSG specificando tipo prodotto, workflow e AOI.

Funzionalità principali

- Parsing avanzato dei nomi file CSK/CSG e decodifica dei metadati principali.
- Accesso facilitato a quicklook, bande e metadati tramite classi Python.
- Estrazione e conversione dei footprint in geometrie Shapely e GeoDataFrame.
- Calcolo e conversione di quicklook in immagini (JPEG, PNG, ecc.).
- Preprocessamento automatico tramite SNAP-GPT con supporto per AOI, parametri multilook stimati e proiezione custom.
- Utility per batch processing e integrazione in pipeline di analisi.

Esempio di utilizzo

```
from sensetrack.cosmo.lib import CSKFile
from sensetrack.cosmo.utils import csk_footprint
from sensetrack.cosmo.preprocessing import CSKPreprocessor

# Carica un file HDF5 CSK
csk = CSKFile('path/to/file.h5')
```

```
print(csk)

# Estrai il footprint come poligono
footprint = csk.footprint_polygon

# Esporta il quicklook come immagine
csk.qlk_to_image.save('quicklook.jpg')

# Preprocessamento automatico (esempio)
preproc = CSKPreprocessor(subset, process)
preproc.run('path/to/file.h5', CRS="EPSG:32632")
```

Esecuzione da linea di comando

```
python -m sensetrack.cosmo.preprocessing --product_type CSK --file path/to/file.h5 --workflo
```

Note

- Il modulo richiede le librerie: `h5py`, `numpy`, `shapely`, `Pillow`, `geopandas`.
- Per il preprocessamento tramite SNAP-GPT è necessario che SNAP sia installato e accessibile dal sistema.
- Le classi e funzioni sono pensate per essere integrate in pipeline di elaborazione SAR multi-sensore.

PRISMA Post-Processing and Conversion Submodule (prisma)

Overview

Il sottopacchetto `sensetrack.prs` fornisce strumenti per la conversione di dati PRISMA, facilitando l'integrazione con software GIS e pipeline di analisi.

Struttura del modulo

- `convert.py`
  - Funzioni per:
    - Lettura di file HDF5 PRISMA.
    - Estrazione di metadati e informazioni di prodotto (`get_prisma_info`).
    - Conversione di bande PRISMA (pan, swir, vnir) in GeoTIFF georeferenziati (`prisma_panchromatic_to_gtiff`).
    - Parsing degli argomenti da linea di comando per conversione e info.

Funzionalità principali

- **Estrazione metadati PRISMA**: stampa informazioni chiave (ID prodotto, livello di processing, percentuale nuvolosità, EPSG) da file HDF5 PRISMA.
- **Conversione a GeoTIFF**: esporta bande panchromatiche, SWIR e VNIR da HDF5 PRISMA a file GeoTIFF georeferenziati, mantenendo proiezione e bounding box.
- **CLI**: permette di lanciare conversioni e info direttamente da terminale con opzioni dedicate.

Esempio di utilizzo

Da Python

```
from sensetrack.prs.convert import get_prisma_info, prisma_panchromatic_to_gtiff

# Estrai info da un file PRISMA
get_prisma_info('path/to/file.h5')

# Converte la banda SWIR in GeoTIFF
prisma_panchromatic_to_gtiff('path/to/file.h5', band='swir')
```

Da terminale

```
python sensetrack/prs/convert.py -f path/to/file.h5 --convert
python sensetrack/prs/convert.py -f path/to/file.h5 --show_info
```

Note

- Le bande supportate per la conversione sono: 'pan', 'swir', 'vnir'.
- L'output GeoTIFF mantiene la georeferenziazione e la proiezione EPSG del prodotto PRISMA.
- Il modulo può essere esteso per supportare altri formati o funzioni di post-processing.

Per dettagli sui parametri e sulle opzioni disponibili, consultare la documentazione inline nei file sorgente.