Generating Level-2 Ocean Products from Sentinel-1 SAR Data

[Ocean Wind Field (OWI)]

Arnob Bormudoi (YU)

Fujitsu Project, Japan

19 Sep-2025

Objective

- To process Sentinel-1 Level-1 Ground Range Detected (GRD) data to generate a higher-level (Level-2) ocean surface wind speed product.
- To leverage GPU acceleration for the computationally intensive parts of the scientific workflow.

From Level-1 to Level-2: Sentinel-1 Ocean Product Generation

Level-2 Product Name	Measures	Required Level-1 Input
Ocean Wind Field (OWI)	Wind Speed	Ground Range Detected (GRD)
Ocean Swell Spectra (OSW)	Wave energy, wavelength, and direction	Single Look Complex (SLC)
Radial Surface Velocity (RVL)	Line-of-sight velocity of the sea surface	Single Look Complex (SLC)

Core Software & Tools

Primary Analysis Code: openwind

Python package specifically designed for estimating high-resolution wind fields from SAR images.

GPU Acceleration Library: CuPy (A key library that provides a NumPy-compatible array interface for running computations on NVIDIA GPUs.)

Source: github.com/nansencenter/openwind

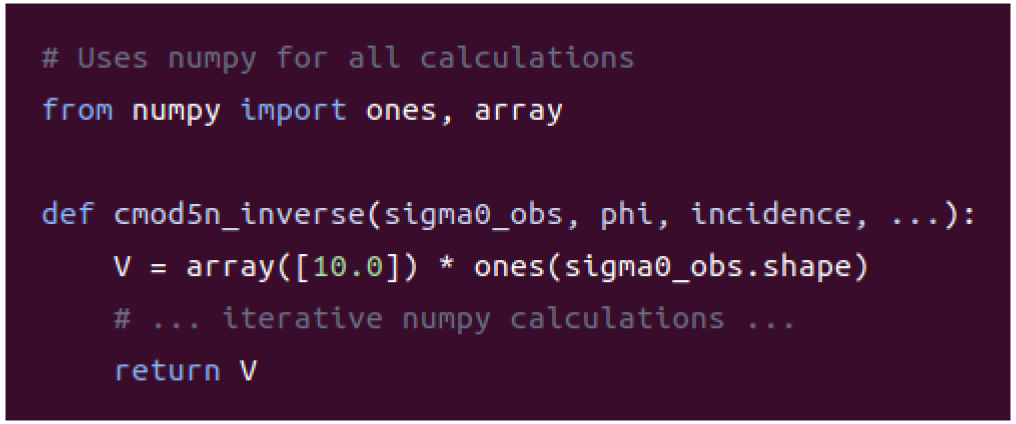
Core Dependency: nansat

The Nansen-Cloud-Satellite Modelling and remote sensing toolbox, which provides the underlying functions for data handling, geolocation, and processing.

Source: github.com/nansencenter/nansat

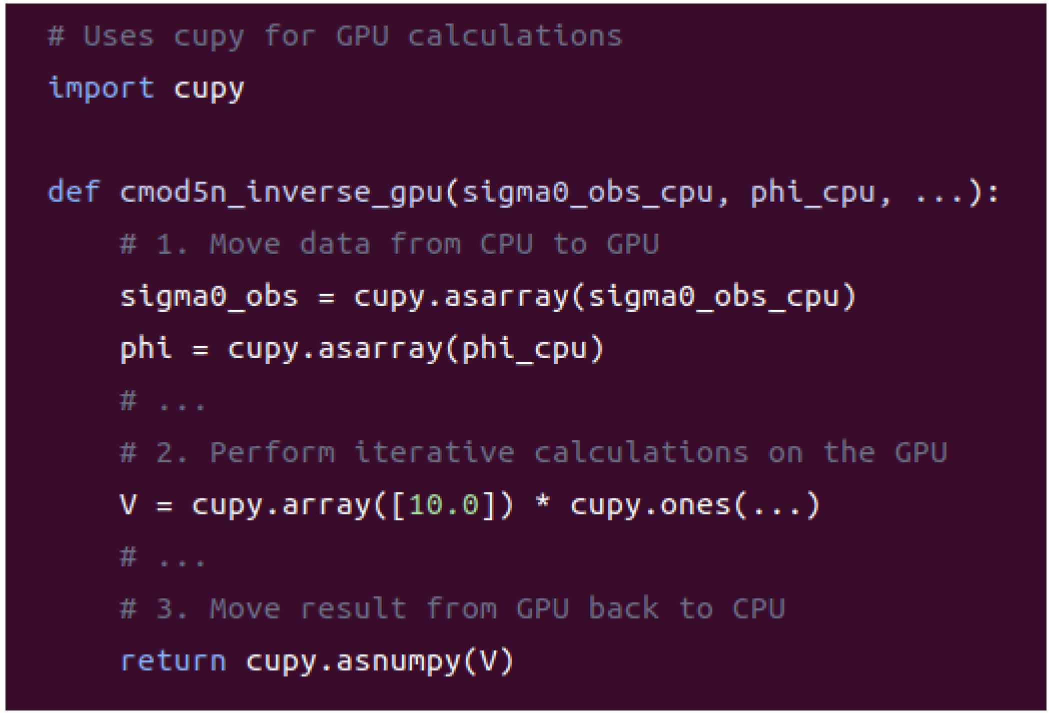
Execution Environment

A dedicated conda environment (py3nansat) was used to manage the complex scientific dependencies (e.g., GDAL, Cartopy, NetCDF4) and cupy-cuda12x package



```python
# Uses numpy for all calculations
from numpy import ones, array

def cmod5n_inverse(sigma0_obs, phi, incidence, ...):
    V = array([10.0]) * ones(sigma0_obs.shape)
    # ... iterative numpy calculations ...
    return V
```



```python
# Uses cupy for GPU calculations
import cupy

def cmod5n_inverse_gpu(sigma0_obs_cpu, phi_cpu, ...):
    # 1. Move data from CPU to GPU
    sigma0_obs = cupy.asarray(sigma0_obs_cpu)
    phi = cupy.asarray(phi_cpu)
    # ...
    # 2. Perform iterative calculations on the GPU
    V = cupy.array([10.0]) * cupy.ones(...)
    # ...
    # 3. Move result from GPU back to CPU
    return cupy.asnumpy(V)
```

The script was modified to import and call the new cmod5n_inverse_gpu function

The Process: From Radar Backscatter to Wind Speed

Input Data

Satellite: Sentinel-1A

Product: Level-1 GRD (Ground Range Detected)

Dataset: S1A_IW_GRDH_1SDV_20250904T204345...SAFE

Geophysical Model: CMOD5n (GPU-Adapted)

An empirical model relating C-band VV-polarized radar backscatter (σ^0) to ocean surface wind speed. The core   calculations were adapted to run on the GPU using the CuPy library.

Execution Script: `openwind/openwind/sar_wind_GPU.py`

This script was modified to import and call our new, custom-built GPU functions

Execution

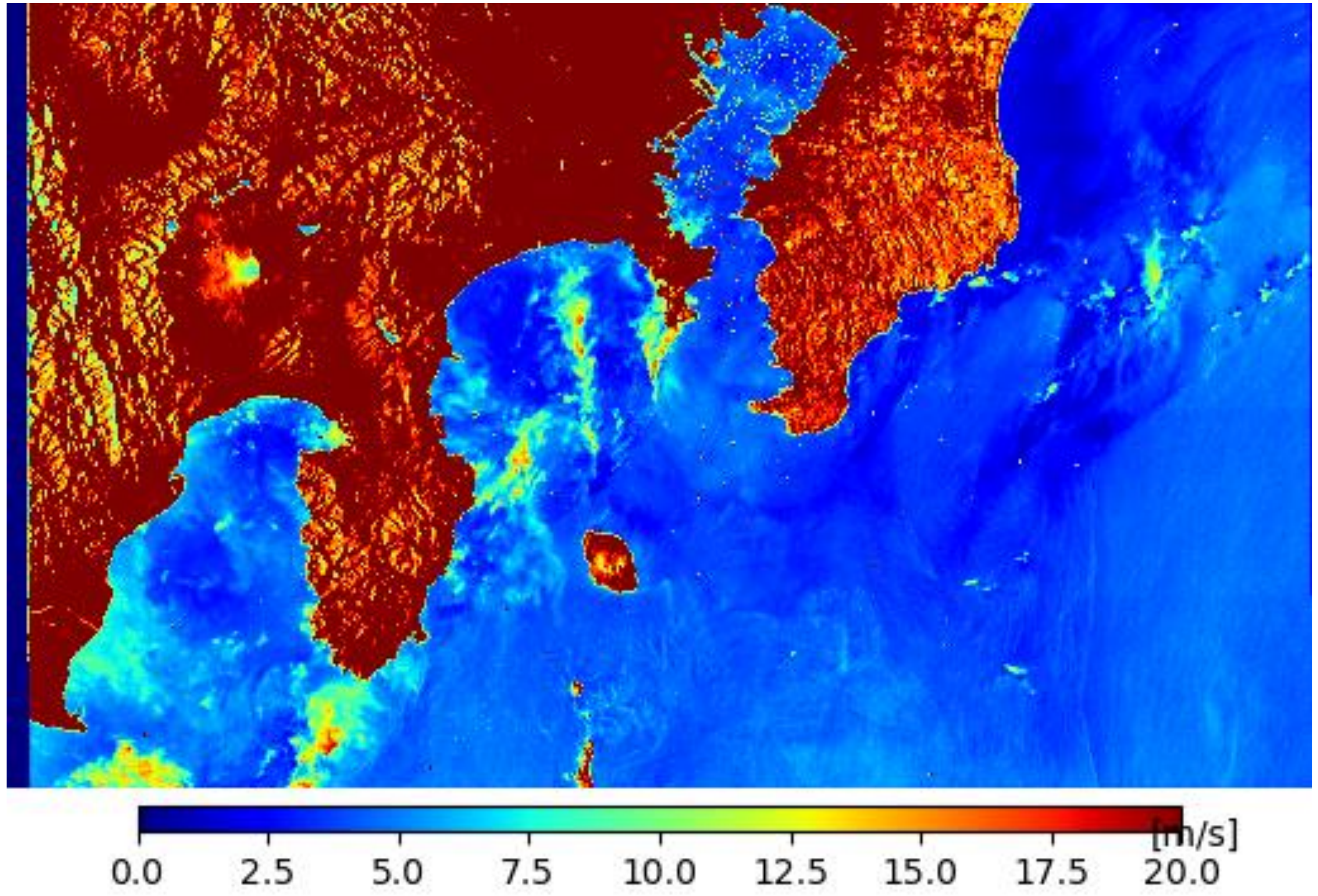
The Python environment was configured by setting the PYTHONPATH to correctly locate the openwind and nansat libraries.

The sar_wind.py script was run, processing the Sentinel-1 data.

Generated Outputs (Level-2 Product):

- ❑ GeoTIFF: tokyo_bay_s1a_wind.tif - A georeferenced data file containing the final wind speed values.
- ❑ PNG: tokyo_bay_s1a_wind.png - A visual representation of the wind field over the target area.

Result



Processing time: 1 minute - 21.7 seconds

The CMOD5n wind model is specifically designed to work over the ocean. It calculates wind speed based on how the roughness of the water surface (caused by wind) scatters the radar signal.

Land surfaces reflect the radar signal much more strongly than the ocean. When the script processes the land pixels, the CMOD5n model misinterprets this strong reflection as being caused by extremely high winds over water.