# pycoxmunk: A python package for computing sea surface reflectance

Simon R. Proud ⓘ [1]

1 National Centre for Earth Observation, RAL Space, STFC Rutherford Appleton Laboratory, Harwell, OX11

Summary

Knowledge of how the sea surface reflects incoming sunlight is of key importance within satellite remote sensing of the Earth. For example, estimating the amount of dust or pollution in the atmosphere requires precise knowledge of the surface reflectance in order to be accurate. This paper introduces *pycoxmunk*, a python library that computes both the sea surface reflectance and the so-called Bidirectional Reflectance Distribution Function (BRDF) across the visible and near-infrared spectrum. Pycoxmunk is designed to work with data from many of the most commonly used satellite sensors and provides output that can be used within retrieval algorithms or as a standalone product in its own right.

Statement of need

The amount of sunlight reflected by a water surface depends upon factors such as the wavelength of light being examined, the wind speed - which affects wave height and the presence of *white caps* atop the waves - and the presence of pigments such as chlorophyll. Satellite sensors typically look down from space towards the water surface and hence have a good view of surface reflectance if we assume clear sky conditions. However, the presence of clouds or aerosols (smoke, dust, etc) can affect the measured *top of atmosphere* reflectance. There exist a multitude of algorithms to detect and analyse these cloud and aerosol features, and they typically require accurate information on the surface reflectance in order to retrieve the atmospheric properties.

Therefore, a tool to calculate the expected sea surface reflectance for a given set of satellite measurements and weather conditions can provide a valuable input into such retrieval algorithms.

State of the field

Previously, most retrieval algorithms have their own inbuilt method of computing this reflectance via a wide range of approaches and algorithms. A particularly well-used approach is that first described by Cox and Munk (Cox, 1954; Cox & Munk, 1954), and expanded upon by subsequent research (Sayer et al., 2010). For example, the ORAC algorithm that can retrieve both cloud and aerosol properties uses Cox-Munk at the core of its surface reflectance simulation (A. C. Povey, 2022; Poulsen et al., 2012). However, both ORAC and other projects have their implementations of Cox-Munk tightly bound with the rest of their codebase, meaning that users cannot easily run just the sea surface reflectance calculation. Many remaining tools, such as the commonly-used HydroLight algorithm that requires a purchased license to function, are therefore also of limited accessibility. An overview of some alternative models is given in (Zhang & Wang, 2010), showing both the limited availability of suitable reflectance models and the good performance of the Cox-Munk technique.

The library introduced here, *pycoxmunk*, implements the Cox-Munk method in a python library that is widely accessible without relying on custom code unique to a given retrieval system such as ORAC. This enables a more widely applicable version of Cox-Munk than has previously been available. In writing this library, ORAC's fortran implementation of Cox-Munk was used as a reference to ensure *pycoxmunk* produced correct output.

Algorithm details

The sea surface has an intrinsic reflectance that depends on how much light is absorbed on the surface and reflected deeper within the water. This primarily depends on the chlorophyll content. At present, *pycoxmunk* assumes a fixed fraction of clorophyll and hence a fixed intrinsic reflectance at a given wavelength that in calm conditions, depends on the viewing geometry:

θ_s: The angle between the sun and vertical such that an angle of 0° specifies that the sun is directly above the target and an angle of 90° specifies that the sun is at the horizon.

θ_v: The angle between the satellite and vertical, defined as above.

ϕ_s: The angle between the sun and North, defined such that 0° and 180° specify that the sun is directly North and directly South of the target respectively and likewise 90° and 270° specify the sun being due East and West respectively.

ϕ_v: The angle between the satellite and North, defined as above.

Where the sun and satellite geometry is favourably aligned, on opposing sides of the target, the sea acts like a mirror, known as specular reflection, and there is a large peak in sea surface reflectance in the *sun glint* region - as shown in Fig 1b. In other geometrical conditions the reflectance is significantly lower.

In windy conditions, the assumption of specular reflection breaks down due to wave activity that, for high wind speeds, generates white caps atop the waves that appear very bright. *pycoxmunk* accounts for white caps using a simple relation(Sayer et al., 2010):

$$\rho_{wc} = R_{wc} \cdot 2.951e^{-6} \cdot v_{wind}^{3.52}$$

Where R_{wc} is the predefined whitecap reflectance at a given wavelength and v_{wind} is the wind speed.

Usage and integration with other libraries

pycoxmunk is designed to work in tandem with the *satpy* library (Raspaud et al., 2018) that reads, calibrates and provides geometry information for many satellite sensors across a wide range of use cases. Once a user has loaded satellite data with satpy they can then pass satpy's main class, the *Scene*, to *pycoxmunk* along with optional arguments to define the user-required processing. *pycoxmunk* will then compute the sea surface reflectance and store the results as additional datasets within the *Scene*, which are directly accessible to the user. A key feature of satpy, which *pycoxmunk* builds upon, is the use of *dask* arrays (Rocklin, 2015) to facilitate processing of datasets larger than the user's total system memory. Internally, all of *pycoxmunk*'s processing is done via dask. If computing the BRDF parameters, *pycoxmunk* will use the *numba* library (Lam et al., 2015) to speed up computation of the *rho_0d*, *rho_0v* and *rho_dd* terms.

To use *pycoxmunk*, a user first needs to load their satellite data via satpy, then pass this to *pycoxmunk*'s main class, *PyCoxMunk*, along with a list of sensor channels to be processed. Optionally, the user can then supply wind information in the U and V directions, preferably at 10m height above the surface. These can be passed as either two floats (for constant wind across the satellite image) or as arrays of equal size to the satellite data for the case of

per-pixel wind data. The user can also supply 'masks' that remove areas of no interest, such as land pixels and cloud filled portions of the image. Lastly, the user calls `retr_coxmunk_refl` to compute the final reflectances, which can then be used by accessing the `PyCoxMunk.scn`.
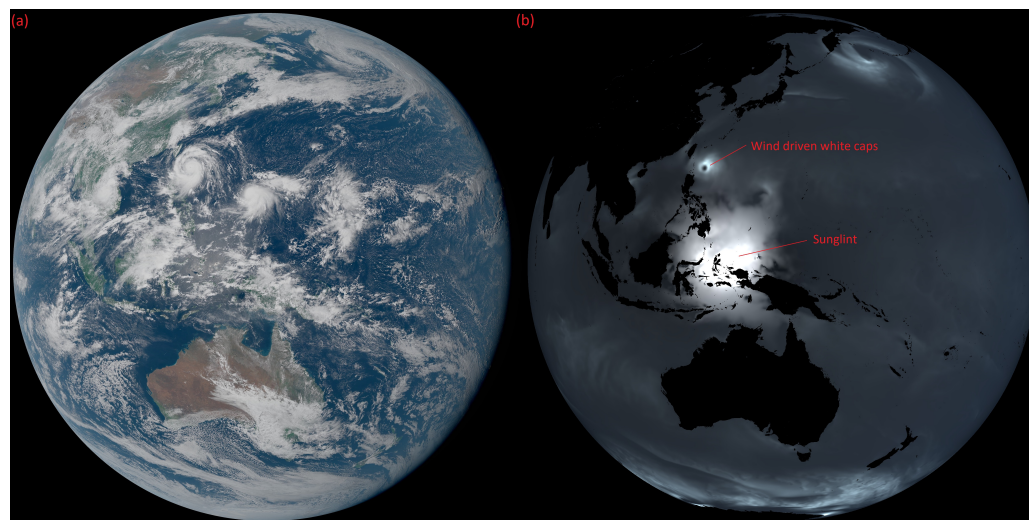
Figures

Figure 1: (a) An image from Japan's Himawari-8 satellite at 04:10 UTC on 13th Sept 2016, Super Typhoon Meranti is visible in the upper-center. (b) The corresponding Cox-Munk surface reflectance for the 0.47, 0.51 and 0.64 micron channels, showing regions of sunglint and wind driven white caps.

Acknowledgements

References

A. C. Povey, S. R. P., G. McGarragh. (2022). ORAC: Optimal retrieval of aerosol and cloud. In *GitHub repository*. GitHub. https://github.com/ORAC-CC/orac

Cox, C. (1954). Statistics of the sea surface derived from sun glitter. *J. Mar. Res.*, *13*, 198–227.

Cox, C., & Munk, W. (1954). Measurement of the roughness of the sea surface from photographs of the sun's glitter. *Josa*, *44*(11), 838–850. https://doi.org/10.1364/josa.44.000838

Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A llvm-based python jit compiler. *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 1–6.

Poulsen, C., Siddans, R., Thomas, G., Sayer, A., Grainger, R., Campmany, E., Dean, S., Arnold, C., & Watts, P. (2012). Cloud retrievals from satellite data using optimal estima-

tion: Evaluation and application to ATSR. *Atmospheric Measurement Techniques*, *5*(8), 1889–1910. https://doi.org/10.5194/amt-5-1889-2012

Raspaud, M., Hoese, D., Dybbroe, A., Lahtinen, P., Devasthale, A., Itkin, M., Hamann, U., Rasmussen, L. Ø., Nielsen, E. S., Leppelt, T., & others. (2018). Pytroll: An open-source, community-driven python framework to process earth observation satellite data. *Bulletin of the American Meteorological Society*, *99*(7), 1329–1336. https://doi.org/10.1175/bams-d-17-0277.1

Rocklin, M. (2015). Dask: Parallel computation with blocked algorithms and task scheduling. *Proceedings of the 14th Python in Science Conference*. https://doi.org/10.25080/majora-7b98e3ed-013

Sayer, A., Thomas, G., & Grainger, R. (2010). A sea surface reflectance model for (A)ATSR, and application to aerosol retrievals. *Atmospheric Measurement Techniques*, *3*(4), 813–838. https://doi.org/10.5194/amt-3-813-2010

Zhang, H., & Wang, M. (2010). Evaluation of sun glint models using MODIS measurements. *Journal of Quantitative Spectroscopy and Radiative Transfer*, *111*(3), 492–506. https://doi.org/10.1016/j.jqsrt.2009.10.001