# CoastSeg: an accessible and extendable hub for coastal shoreline detection and mapping

06 February 2024

## Summary

`CoastSeg` is an interactive browser-based program consisting of jupyter notebooks and python scripts that aims to broaden the adoption of satellite-based shoreline detection (SDS) and coastal landcover mapping workflows among coastal scientists and practioners. SDS is a sub-field of coastal sciences that aims to detect and post-process a time-series of shoreline locations from a time-series of publicly available satellite imagery (Turner et al. 2021; Vitousek et al. 2023). `CoastSeg` is a python package installed via pip into a conda environment. `CoastSeg` also provides jupyter notebooks and python scripts that call functions and classes in the `CoastSeg` core functionalities for specific workflows. All API codes, notebooks, scripts, and documentation are hosted on the `CoastSeg` github repository.

CoastSeg has three broad aims. The first is to be an API consisting of core SDS workflow functionalities such as file input/output, image downloading, geospatial conversion, tidal model API handling, mapping 2d shorelines to 1d transects, and numerous other workflows common to a basic SDS standard, without the implementation of a particular waterline estimation workflow. This workflow is crucial to the success of any SDS workflow because it is the step that identifies the waterline, or the boundary between sea and land. The idea behind CoastSeg designed as an API is so users could extend or customize functionality using scripts and notebooks.

The second aim of CoastSeg is therefore to provide fully functioning SDS implementations in an accessible browser notebook format. Our principal objective to date has been to re-implement and improve upon a popular existing toolbox, `CoastSat` (Vos et al. 2019), allowing the user to carry out the well-established `CoastSat` SDS workflow (Castelle et al. 2021; Vos, Harley, et al. 2023; Vos, Splinter, et al. 2023; Warrick et al. 2023; Konstantinou et al. 2023), but in a more accessible way within the `CoastSeg` platform. In order to acheive this, we created and maintain `CoastSat-package` (Vos and Fitzpatrick 2023), a python package installed via pip into the `CoastSeg` conda environment that contains

re-implemented versions of many of the original `CoastSat` codes. The `CoastSeg` re-implementation of the `CoastSat` workflow is end-to-end within a single notebook. That notebook allows the user to by enabling users to, among other tasks: a) define a region of interest on a webmap and upload geospatial vector format files; b) define, download and post-process satellite imagery; c) identify waterlines in that imagery using the `CoastSat` method (Vos et al. 2019); d) correct those waterlines to elevation-based shorelines using tidal elevation-datum corrections provided through interaction with the pyTMD (Alley et al. 2017) API; and e) download output files in a variety of modern geospatial and other formats for subsequent analysis.

The third and final aim of `CoastSeg` is to implement a method to carry out SDS workflows in experimental and collaborative contexts, which aids both oversight and reproducibility as well as practical needs based on division of labor. We do this using workflow `sessions`, a system that enables users to iteratively experiment with different combinations of settings. `CoastSeg` enables fully reproducible workflows because everything is saved in a way such that a users can share their sessions with others, enabling peers to replicate experiments, build upon previous work, or access data downloaded by someone else. This simplifies handovers to new users from existing users, simplifies teaching of the program, and encourages collective experimentation which may result in better shoreline data.

`CoastSeg` is designed to be extendable, serving as a hub that hosts alternative SDS workflows and similar workflows that can be encoded in a jupyter notebook built upon the `CoastSeg` and `CoastSat-package` core functionalities. Additional notebooks will carry out shoreline extraction and coastal landcover mapping using alternative methods. The first is an alternative SDS workflow based on a deep-learning based semantic segmentation model, that we briefly summarize at the end of this paper. To implement a custom waterline detection workflow, the originator of that workflow would contribute a new jupyter notebook, and also likely contribute to the `CoastSeg` source code to add their specific waterline detection algorithm there, that they could call in their notebook.

## Statement of Need

Coastal scientists and practioners now have access to extensive collections of satellite data spanning more than four decades. However, it's only in recent years that advancements in algorithms, machine learning, and deep learning have enabled the automation of processing this satellite imagery to accurately identify and map shorelines from imagery, a process known as Satellite-Derived Shorelines, or SDS. SDS workflows are gaining rapidly in popularity, and in particular the `CoastSat` workflow for instantaneous SDS, that is, one shoreline extracted per timestamped image, has been widely adopted since its release in 2018 (Vos et al. 2019). Existing open-source software for SDS often require the user to navigate between platforms (non-reproducible elements), develop custom

code, and/or engage in substantial manual effort.

We sought to build a platform that not only allowed the user to adopt the `CoastSat` workflow in a re-implementation than ran in a single jupyter notebook, quicker, and more seamlessly, but also one that facilitates experimentation with the many settings that can govern shoreline accuracy, extent, and number. Further, `CoastSeg` has been designed specifically to host alternative SDS workflows, recognizing that it is a nascent field of coastal science, and the optimal methodologies for all coastal environments and sources of imagery are yet to be established. Therefore `CoastSeg` will provide a means with which to extract shorelines using multiple methods and adopt the one that most suits their needs, or implement a new methof of their own devising on the `CoastSeg` platform.

We summarize the needs met by the `CoastSeg` project as follows:

- A re-implementation of (and improvement of) the CoastSat workflow with pip-installable APIs, and `coastsat-package`.
- A browser-based workflow and an interactive mapping interface provided by Leafmap (Wu 2021).
- A more accessible, entirely graphical and menu-based SDS workflow, with no exposure of source code to the user.
- A session system that makes experimenting to discover the setting that optimal extract shorelines from satellite imagery.
- Improved core SDS workflow components, such as faster and more seamless tidal correction workflow, and faster image downloading.
- Consolidation of workflows in a single platform and reusable codebase.
- An extendable hub of alternative SDS workflows in one location.

# Implementation of core SDS workflow

## Architecture & Design

At a high level, `CoastSeg` is designed to be an accessible and extendible hub for both `CoastSat`-based and alternate workflows, each of which is implemented in a single notebook. The user is therefore presented with a single menu of notebooks, each of which calls on a common set of core functionalities provided by `Coastseg` and `Coastsat--package`, and exporting data to common file formats and conventions.

CoastSeg is installable as a pip package into a conda environment. `CoastSeg` notebooks are accessed from GitHub. We also created a pip package for the `Coastsat` workflow in order to a) improve the `CoastSat` method's software implementation without affecting the parent repository, and b) to install as a pip package into a conda environment, rather than duplicate code from CoastSat.

`CoastSeg` is built with a object-oriented architecture, where elements required by the `CoastSat` workflow such as Regions of Interest, reference shorelines, and transects are represented as distinct classes on the map. Each class stores data specific to that feature type as well as encompassing methods for styling the feature on the map, downloading default features, and executing various post-processing functions.

## Sessions

SDS workflows require settings in order to extract optimal shorelines. There are numerous settings in the `CoastSat` workflow, and sometimes determining optimal shorelines can be an iterative process requiring experimentation with settings. Sub-optimal shoreline extraction may result merely through user fatigue. Therefore, `CoastSeg` employs a `session`-based system that enables users to iteratively experiment with different combinations of settings. Each time the user makes adjustments to the settings used to extract shorelines from the imagery a new session folder is saved with the updated settings. This session system is what makes `CoastSeg` fully reproducible because all the settings, inputs, and outputs are stored within each session as well as a reference to what downloaded data was used to generate the extracted shorelines in the session. Moreover, the session system in "CoastSeg" fosters a collaborative environment. Users can share their sessions with others, enabling peers to replicate experiments, build upon previous work, or access data downloaded by someone else. This simplifies the process for new users and encourages collective experimentation and data sharing. This reproducibility and collaboration are beneficial in research contexts.

## Improvements to the CoastSat workflow

### Accessibility

`CoastSeg` facilitates entirely browser-based workflows with an interactive webmap and ipywidget controls (Figure 1). It interfaces with the Zenodo API to download reference shorelines for any location in the world, organized into 5x5 degree chunks in geoJSON format (Daniel Buscombe 2023) as well as transects, themselves providing beachface slope metadata (Daniel Buscombe and Fitzpatrick 2023) available on hover. We have implemented rigorous error handling using developer log files, user report files, and informative error messages that suggest problem fixes. We have also provided a set of utility scripts for common data input/output tasks, oftem the result of specific requests from our software testers (see Acknowledgements). As well as a project wiki and improved documentation, we have researched min, max, and recommended values for all settings, and have provided visual project management aids.

### Performance

We impelemented faster image retrieval from Google Earth Engine ... more details

We impelemented more stable (near-uninterruptible) image downloading ... more details

We added helpful workflow components such as image filtering options; for example, users can now filter their imagery based on image size and the proportion of no data pixels in image. Additionally, the user can decide to turn off cloud masking, which is necessary when the cloud masking process fails and obscures non-cloudy regions such as bright pixels of sand beaches. Finally, we replaced non-cross-platform components of the original workflow, for example the pickle format was replaced with JSON or geoJSON formats which are both human-readable and compatible with GIS and webGIS.

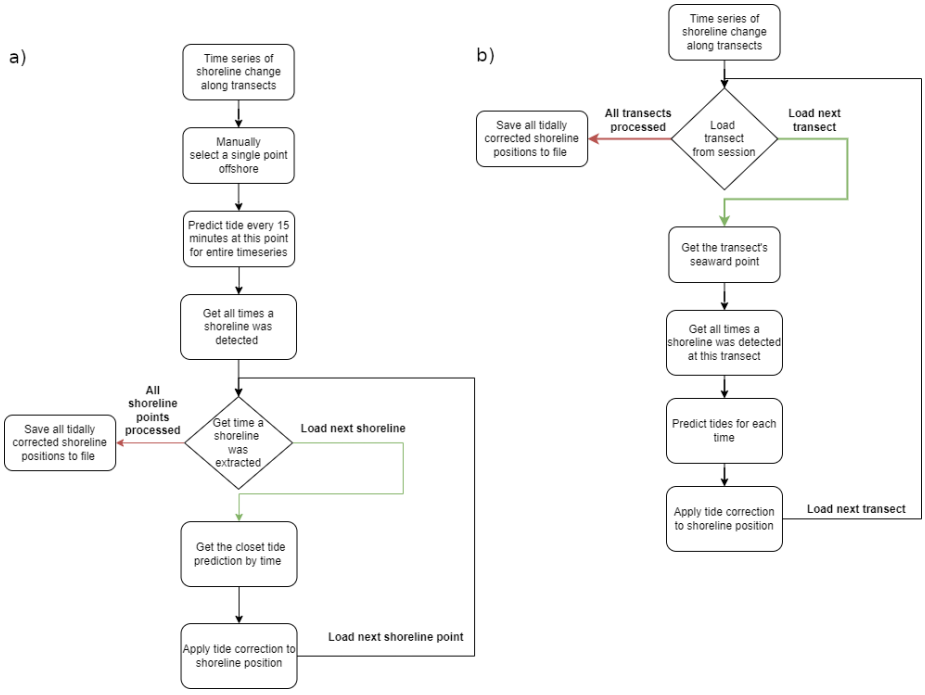


Figure 1: Schematic of the tidal correction workflow used by a) CoastSat and b) CoastSeg.

### Tide

Tidal correction (Figure 2) of shorelines involves estimating the tide height for any location and time using the pyTMD (Alley et al. 2017) API to model the tide. pyTMD provides an accessible script for FES14 (Lyard et al. 2021) tidal model

data access, includes several models other than FES14 including polar-specific models. We created an automated workflow that splits the FES2014 model data into 11 global regions (an idea adopted from DEA-coastlines). This allows the program to access only a subset of the data, facilitating fast tide estimates (in minutes rather than hours for long satellite time-series).

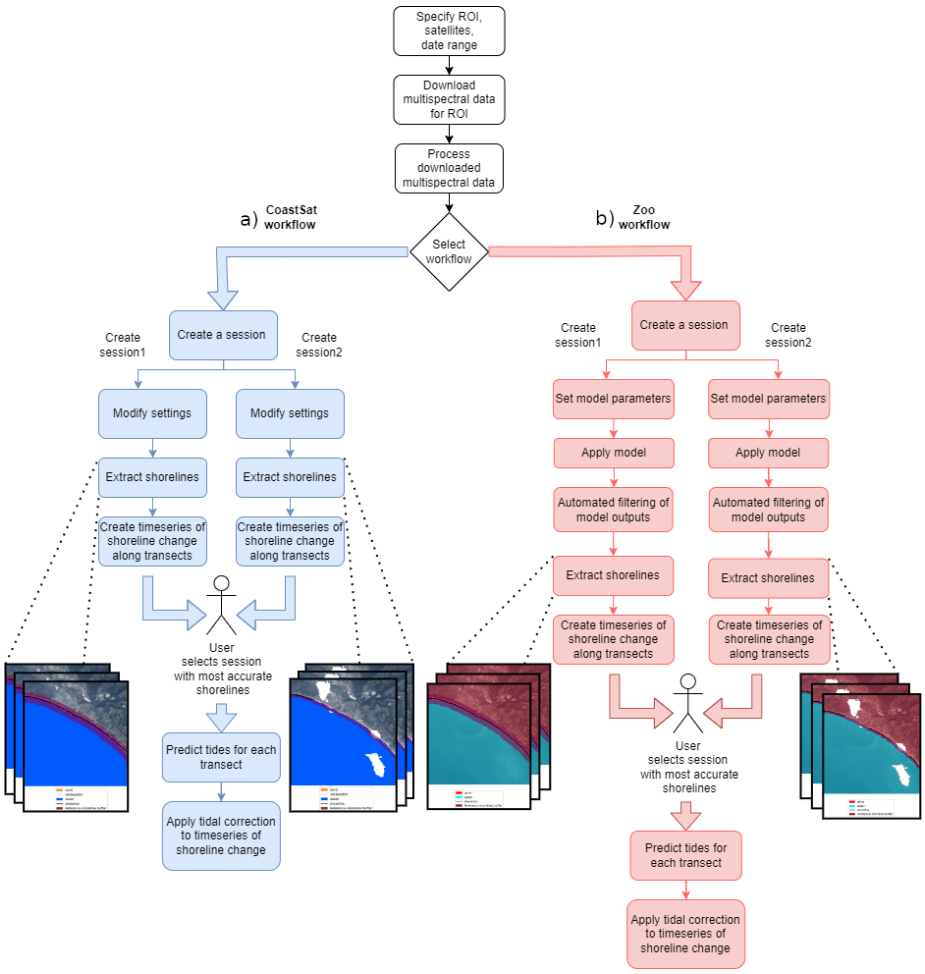


Figure 2: Schematic of the SDS workflows currently available in CoastSeg. a) CoastSat workflow; b) Zoo workflow.

## Implementation of an Alternative Deep-Learning-Based SDS Workflow

As we noted above, we have developed a notebook that carries out an alternative SDS workflow based on a deep-learning based semantic segmentation model. To implement this custom workflow, we created a new jupyter notebook, and added source code to the `CoastSeg` API. The changes ensured that the inputs and outputs were those expected by the `CoastSeg` core API. We call this alternative workflow the `Zoo` workflow, in reference to the fact that the deep learning models implemented originate from the `Segmentation Zoo` github repository, and result from the `Segmentation Gym` deep-learning based image segmentation model training package (Buscombe and Goldstein 2022). Figure 3 describes in detail how the two workflows differ.

## Acknowledgements

The author would like to thank Qiusheng Wu, developer of `Leafmap`, whichs adds a lot of functionality to `CoastSeg`. Thanks also to the developers and maintainers of pyTMD, DEA-tools, xarray, and GDAL, without which this project would be impossible. We acknowledge contributions from Robbi Bishop-Taylor, Evan Goldstein, Venus Ku, software testing and suggestions from Eli Lazarus, Andrea O'Neill, Ann Gibbs, Kathryn Weber, and Julia Heslin, and support from USGS Coastal Hazards and Resources Program, and Merbok Supplemental.

## References

Alley, K, K Brunt, S Howard, L Padman, M Siegfried, and T Sutterly. 2017. "PyTMD: Python Based Tidal Prediction Software." doi:%2010.5281/zenodo.5555395.

Buscombe, Daniel. 2023. "CoastSeg: Shoreline data at 30-m spatial resolution for 5x5 degree regions of the world, in geoJSON format." Zenodo. https://doi.org/10.5281/zenodo.7786276.

Buscombe, Daniel, and Sharon Fitzpatrick. 2023. "CoastSeg: Beach transects and beachface slope database v1.0." Zenodo. https://doi.org/10.5281/zenodo.8187949.

Buscombe, D., and E. Goldstein. 2022. "A Reproducible and Reusable Pipeline for Segmentation of Geoscientific Imagery." *Earth and Space Science* 9 (9): e2022EA002332.

Castelle, B, G Masselink, T Scott, C Stokes, A Konstantinou, V Marieu, and S Bujan. 2021. "Satellite-Derived Shoreline Detection at a High-Energy Meso-Macrotidal Beach." *Geomorphology* 383: 107707.

Konstantinou, A, T Scott, G Masselink, K Stokes, D Conley, and B Castelle. 2023. "Satellite-Based Shoreline Detection Along High-Energy Macrotidal Coasts and Influence of Beach State." *Marine Geology*, 107082.

Lyard, FH, DJ Allain, M Cancet, L Carrere, and N Picot. 2021. "FES2014 Global Ocean Tide Atlas: Design and Performance." *Ocean Science* 17 (3): 615–49.

Turner, IL, MD Harley, R Almar, and EWJ Bergsma. 2021. "Satellite Optical Imagery in Coastal Engineering." *Coastal Engineering* 167: 103919.

Vitousek, S, D Buscombe, K Vos, PL Barnard, AC Ritchie, and JA Warrick. 2023. "The Future of Coastal Monitoring Through Satellite Remote Sensing." *Cambridge Prisms: Coastal Futures* 1: e10.

Vos, K, and S Fitzpatrick. 2023. "Coastsat-Package."

Vos, K, MD Harley, IL Turner, and KD Splinter. 2023. "Pacific Shoreline Erosion and Accretion Patterns Controlled by El Niño/Southern Oscillation." *Nature Geoscience* 16 (2): 140–46.

Vos, K, KD Splinter, MD Harley, JA Simmons, and IL Turner. 2019. "CoastSat: A Google Earth Engine-enabled Python toolkit to extract shorelines from publicly available satellite imagery." *Environmental Modelling & Software* 122: 104528.

Vos, K, KD Splinter, J Palomar-Vázquez, JE Pardo-Pascual, J Almonacid-Caballer, C Cabezas-Rabadán, EC Kras, et al. 2023. "Benchmarking Satellite-Derived Shoreline Mapping Algorithms." *Communications Earth & Environment* 4 (1): 345.

Warrick, JA, K Vos, D Buscombe, AC Ritchie, and JA Curtis. 2023. "A Large Sediment Accretion Wave Along a Northern California Littoral Cell." *Journal of Geophysical Research: Earth Surface*, e2023JF007135.

Wu, Q. 2021. "Leafmap: A Python Package for Interactive Mapping and Geospatial Analysis with Minimal Coding in a Jupyter Environment." *Journal of Open Source Software* 6 (63): 3414.