CoastSeg: an accessible and extendable hub for satellite-derived-shoreline (SDS) detection and mapping

Sharon Fitzpatrick [1], **Daniel Buscombe** [1], **Jonathan A. Warrick** [2], **Mark A. Lundine** [2], **and Kilian Vos** [3]

1 Contracted to U.S. Geological Survey Pacific Coastal and Marine Science Center, Santa Cruz, California, United States. **2** U.S. Geological Survey Pacific Coastal and Marine Science Center, Santa Cruz, California, United States. **3** New South Wales Department of Planning and Environment, Sydney, Australia

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

DRAFT

Summary

`CoastSeg` is an interactive browser-based program that aims to broaden the adoption of satellite-derived shoreline (SDS) detection and coastal landcover mapping workflows among coastal scientists and coastal resource management practitioners. SDS is a sub-field of coastal sciences that aims to detect and post-process a time-series of shoreline locations from publicly available satellite imagery (Turner et al., 2021; Vitousek, Buscombe, et al., 2023). `CoastSeg` is a Python package installed via pip into a conda environment that serves as an toolkit for building custom SDS workflows. `CoastSeg` also provides full SDS workflow implementations via Jupyter notebooks and Python scripts that call functions and classes in the core `CoastSeg` toolkit for specific workflows. CoastSeg provides two fully functioning SDS workflows and its design allows for collaborators in the SDS software community to contribute additional workflows. All the codes, notebooks, scripts, and documentation are hosted on the `CoastSeg` GitHub repository.

So-called 'instantaneous' SDS workflows, where shorelines are extracted from each individual satellite image rather than temporal composites (Bishop-Taylor et al., 2021), follow a basic recipe, namely 1) waterline estimation, where the 2D (x,y) location of the land-sea interface is determined, and 2) water-level correction, where the waterline location is mapped onto a shore-perpendicular transect, converted to a linear distance along that transect, then corrected for water level, and referenced to a particular elevation contour on the beach (Vos et al., 2019). The resulting measurement is called a 'shoreline' and it is the location that the waterline intersects a particular elevation datum. Water level corrections typically only account for tide (Vos et al., 2019) but recently SDS workflows have incorporated both wave setup and runup correction, which are a function of the instantaneous wave field at the time of image acquisition (Castelle et al., 2021; Konstantinou et al., 2023; Vitousek, Buscombe, et al., 2023; Vitousek, Vos, et al., 2023).

`CoastSeg` has three broad aims. The first aim is to be an toolkit consisting functions that operate the core SDS workflow functionalities. This includes file input/output, image downloading, geospatial conversion, tidal model API handling, mapping 2D shorelines to 1D transect-based measurements, and numerous other functions common to a basic SDS workflow, regardless of a particular waterline estimation methodology. This waterline detection algorithm will be crucial to the success of any SDS workflow because it is the step that identifies the the boundary between sea and land which serves as the basis for shoreline mapping. The idea behind the design of `CoastSeg` is that 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 with a well-supported literature (Castelle et al., 2021, 2022; Konstantinou et al., 2023; McLean et al., 2023; Vandenhove et al., 2024; Vitousek, Vos, et al., 2023; Vos, Harley, et al., 2023; Vos, Splinter, et al., 2023; Warrick et al., 2023), but in a more accessible and convenient way within the CoastSeg platform. In order to achieve this, we developed CoastSat-package (Vos & Fitzpatrick, 2023), a Python package that is installed into the CoastSeg conda environment. CoastSat-package contains re-implemented versions of the original CoastSat codes, addresses the lack of pip or conda installability of CoastSat, and isolates the CoastSeg-specific enhancements from the original CoastSat code. The CoastSeg re-implementation of the CoastSat workflow is end-to-end within a single notebook. That notebook allows the user to, among other tasks: a) define a Region of Interest (ROI) 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) save output files in a variety of modern geospatial and other formats for subsequent analysis. Additionally, CoastSeg's toolkit-based design enables it to run as non-interactive scripts, catering to larger scale shoreline analysis projects.This flexibility ensures that CoastSeg can accommodate a wide range of research needs, from detailed, interactive exploration to extensive, automated analyses.

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 sessions, a mechanism for saving the current state of the application into a session's folder. This folder contains all necessary inputs, outputs, and references to downloaded data used to generate the results. Sessions allow users to iteratively experiment with different combinations of settings and makes CoastSeg fully reproducible because everything needed to reproduce the session is saved to the folder. 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 also 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 can be designed to carry out shoreline extraction and coastal landcover mapping using alternative methods. We provide an example of an alternative SDS workflow based on a deep-learning based semantic segmentation model that is briefly summarized at the end of this paper. To implement a custom waterline detection workflow the originator of that workflow would contribute new Jupyter notebook, and add their specific waterline detection algorithm to the CoastSeg source code, so it could be used in their notebook's implementation.

Statement of Need

Coastal scientists and resource managers 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 (Almonacid-Caballer et al., 2016; Garcia-Rubio et al., 2015) are gaining rapidly in popularity, and in particular since the publication of the open-source implementation of the CoastSat workflow (Vos, 2023) for instantaneous SDS in 2018 (Vos et al., 2019). Existing open-source software for SDS often require the user to

DRAFT

navigate between platforms (non-reproducible elements), develop custom code, and/or engage in substantial manual effort.

We built CoastSeg with the aim of enhancing the CoastSat workflow. Our design streamlines the entire shoreline extraction process, thus facilitating a more efficient experimental approach to determine the optimal combination of settings to extract the greatest number of accurate shorelines. CoastSeg achieves these improvements through several key advancements: it ensures reproducible sessions for consistent comparison and analysis; introduces additional filtering mechanisms to refine results; and provides an interactive user webmap that allows the users to users to view the quality of the extracted shorelines. 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` provides a means with which to extract shorelines using multiple methods and adopt the one that most suits their needs, or implement a new methods.

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 (mandatory) exposure of source code to the user.
- A session system that streamlines the experimentation process to find the settings that extract optimal shorelines from satellite imagery.
- Improved core SDS workflow components, such as a 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 extendable 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 package into a conda environment. `CoastSeg` notebooks are accessed from GitHub. We also created a pip package for the `CoastSat` workflow we named `CoastSat-package` in order to a) improve the `CoastSat` method's software implementation without affecting the parent repository, and b) to install as a 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 objects 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 manipulating various 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 or a combination of misconfigured settings. 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. It interfaces with the Zenodo API to download reference shorelines for any location in the world, organized into 5x5 degree chunks in GeoJSON format (Buscombe, 2023) as well as transects, themselves providing beachface slope metadata (Buscombe & Fitzpatrick, 2023) available when users hover over each transect with their cursor. We have improved the reliability of `CoastSeg` through rigorous error handling, which includes developer log files for in-depth diagnostics, user report files for transparency, and detailed error messages that provide guidance for troubleshooting and problem resolution. We have also provided a set of utility scripts for common data input/output tasks, often the result of specific requests from our software testers (see Acknowledgments). In addition to a project wiki and improved documentation, we have researched minimum, maximum, and recommended values for all settings, set suggested default values, and have provided visual project management aids.

Performance

`CoastSeg` improves upon the Google Earth Engine-based image retrieval process adopted by `CoastSat` by offering a more reliable and efficient download mechanism. Like `CoastSat`, we limit image sources to only the Landsat and Sentinel missions, which are publicly available to all. `CoastSeg` supports downloading multiple regions of interest in a single session, and ensures downloads persist even over an unstable internet connection. This is important because SDS users typically download all available imagery from an ROI, which may amount to several hundred to thousand individual downloaded scenes. Should a download error occur, `CoastSeg` briefly pauses before reconnecting to Google Earth Engine, ensuring the process doesn't halt completely. In cases where image downloading fails repeatedly, the filename is logged to a report file located within the downloaded data folder. This report file tracks the status of all requested images from Google Earth Engine. `CoastSeg`'s reliable image retrieval process enhances coastal monitoring by facilitating easier data management and collaboration.

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 an 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.

The Journal of Open Source Software

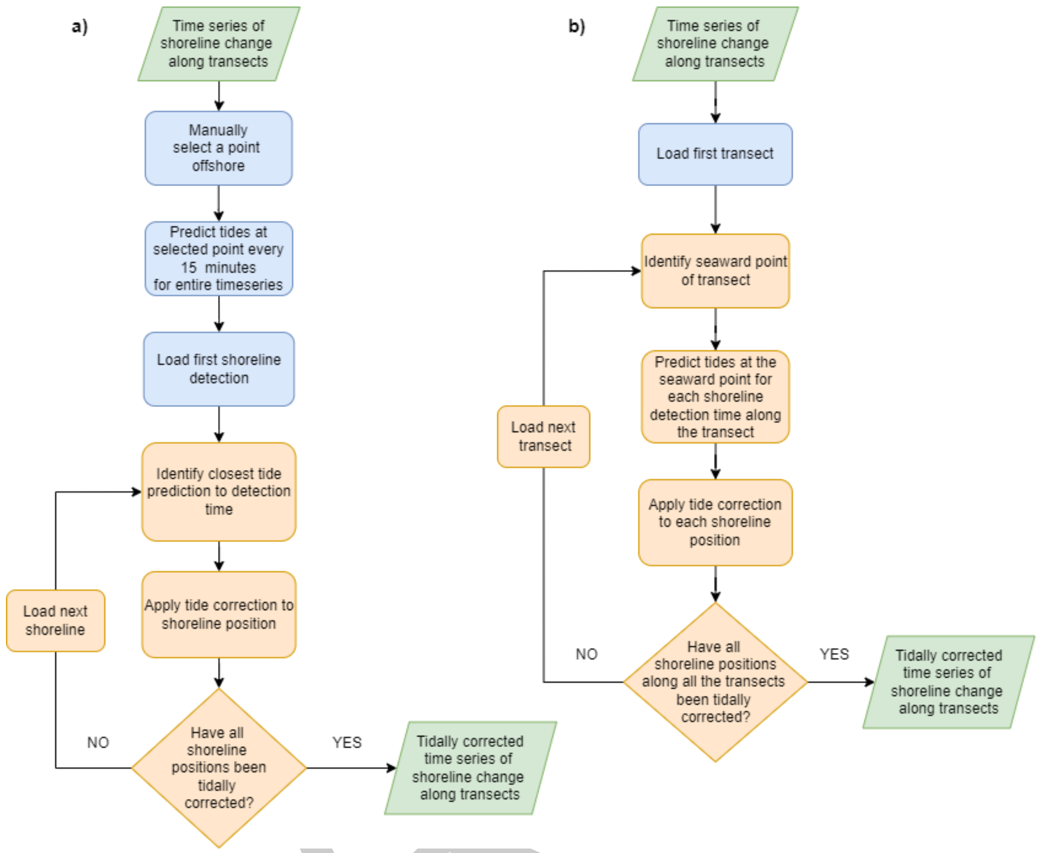

Figure 1: Schematic of the tidal correction workflow used by a) `CoastSat` and b) `CoastSeg`.

DRAFT

Tide

The CoastSat methodology for applying tide correction to shoreline positions involved a multi-step process. First the user would need to independently download and configure the FES14 (Lyard et al., 2021) tide model,a widely recognized tidal model. After configuring the tide model, users would then generate tide estimates at 15-minute intervals for a single location within their ROI across the entire satellite imagery time series. The tide estimate closest to the time of shoreline detection was used to adjust the shoreline position. This method, while comprehensive, was time-consuming, potentially requiring hours to generate all necessary tide estimates.

In contrast, CoastSeg introduces a significant improvement to this process by leveraging the pyTMD API (Alley et al., 2017) for a more streamlined and accurate approach to tidal correction (Figure 1). pyTMD facilitates downloading a variety of tide models, including FES14 and models specific to polar regions, and automates tide estimations. We provide an automated workflow that downloads and subdivides the FES2014 model data into 11 global regions (an idea adopted from (Krause et al., 2021)). This subdivision allows the program to access only relevant subsets of data, drastically reducing the time required to estimate tides—from hours to minutes for multi-decadal satellite time series. Furthermore, CoastSeg calculates tide estimates for each transect corresponding to the times shorelines were detected. This ensures tide corrections are based on temporal and spatial matches, enhancing the accuracy of shoreline position adjustments.

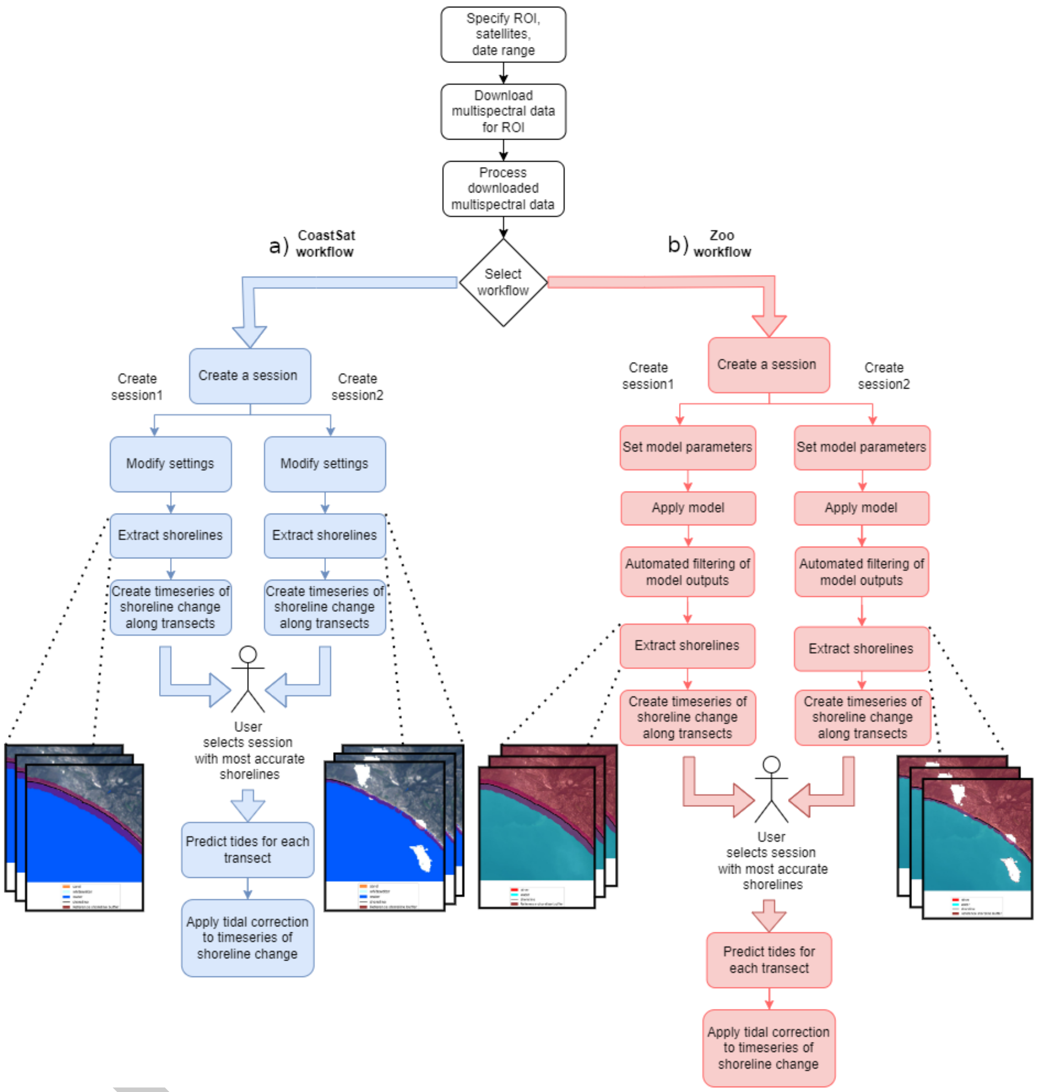

Figure 2: Schematic of the SDS workflows currently available in `CoastSeg`. a) `CoastSat` workflow; b) `Zoo` workflow.

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 deep-learning based semantic segmentation models. The name 'CoastSeg' is derived from this functionality—using semantic segmentation models for the precise classification of coastal geomorphological features. This advanced classification refines the extraction of shoreline data from satellite imagery. To implement this custom workflow, we created a new Jupyter notebook, and added source code to the `CoastSeg` codebase. The changes ensured that the inputs and outputs were those expected by core functions in `CoastSeg` toolkit. 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 & Goldstein, 2022). The name 'Zoo' has become a standard for online trained ML models (NVIDIA, 2023; PyTorch, 2020), and the repository contains both SDS models and others. Figure 2 describes in detail how the two workflows differ. The optimal

SDS workflow adopted for waterline detection, as determined against field validation data, will be the subject of a future manuscript.

Project Roadmap

We intend CoastSeg to be a collaborative research project and encourage contributions from the SDS community. As well as implementing alternative SDS waterline detection workflows, other improvements that could continue to be made include more (or more refined) outlier detection methods, image filtering procedures, and other basic image pre- or post-processing routines, especially image restoration on degraded imagery (Vitousek, Buscombe, et al., 2023). Such additions would all be possible without major changes to the existing CoastSeg toolkit.

Integration of new models for the deep-learning workflow are planned, based on non-dimensionalized water index (NDWI) and modified non-dimensionalized water index (MNDWI) spectral indices, as is a new CoastSeg toolbox extension for daily 3-m Planetscope imagery (Doherty et al., 2022) from Planet Labs (Planet Labs, 2018). Docker may be adopted in the future for managing dependencies in the conda virtual environment required to run the program. Other sources of imagery and other spectral indices may have value in SDS workflows, and we encourage SDS users to contribute their advances through a CoastSeg Jupyter notebook implementation.

It would be also be possible to incorporate automated satellite image subpixel co-registration in CoastSeg using the AROSICS package (Scheffler et al., 2017). This would co-register all available imagery to the nearest-in-time LandSat image. Further, future work could include accounting for the contributions of runup and setup to total water level (Vitousek, Vos, et al., 2023; Vos, Splinter, et al., 2023). In practice, this would merely add/subtract a height from the instantaneous predicted tide, then apply horizontal correction. However, the specific methods used to estimate runup or setup from the prevailing wave field would require integration with observed or hindcasted databases of wave conditions.

Acknowledgments

The authors would like to thank Qiusheng Wu, developer of Leafmap, which 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 Catherine Janda, Eli Lazarus, Andrea O'Neill, Ann Gibbs, Rachel Henderson, Emily Himmelstoss, Kathryn Weber, and Julia Heslin, and support from USGS Coastal Hazards and Resources Program, and USGS Merbok Supplemental.

Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

References

Alley, K., Brunt, K., Howard, S., Padman, L., Siegfried, M., & Sutterly, T. (2017). *PyTMD: Python based tidal prediction software*. https://doi.org10.5281/zenodo.5555395; Zenodo.

Almonacid-Caballer, J., Sanchez-Garcia, E., Pardo-Pascual, J. E., Balaguer-Beser, A. A., & Palomar-Vazquez, J. (2016). Evaluation of annual mean shoreline position deduced from Landsat imagery as a mid-term coastal evolution indicator. *Marine Geology*, *372*, 79–88.

Bishop-Taylor, R., Nanson, R., Sagar, S., & Lymburner, L. (2021). Mapping australia's dynamic coastline at mean sea level using three decades of landsat imagery. *Remote Sensing of Environment*, *267*, 112734.

DRAFT

Buscombe, D. (2023). *CoastSeg: Shoreline data at 30-m spatial resolution for 5x5 degree regions of the world, in geoJSON format.* (Version v1.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.7786276

Buscombe, D., & Fitzpatrick, S. (2023). *CoastSeg: Beach transects and beachface slope database v1.0* (Version v1.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.8187949

Buscombe, D., & Goldstein, E. (2022). A reproducible and reusable pipeline for segmentation of geoscientific imagery. *Earth and Space Science*, *9*(9), e2022EA002332.

Castelle, B., Masselink, G., Scott, T., Stokes, C., Konstantinou, A., Marieu, V., & Bujan, S. (2021). Satellite-derived shoreline detection at a high-energy meso-macrotidal beach. *Geomorphology*, *383*, 107707.

Castelle, B., Ritz, A., Marieu, V., Lerma, A. N., & Vandenhove, M. (2022). Primary drivers of multidecadal spatial and temporal patterns of shoreline change derived from optical satellite imagery. *Geomorphology*, *413*, 108360.

Doherty, Y., Harley, M. D., Vos, K., & Splinter, K. D. (2022). A Python toolkit to monitor sandy shoreline change using high-resolution PlanetScope cubesats. *Environmental Modelling & Software*, *157*, 105512.

Garcia-Rubio, G., Huntley, D., & Russell, P. (2015). Evaluating shoreline identification using optical satellite images. *Marine Geology*, *359*, 96–105.

Konstantinou, A., Scott, T., Masselink, G., Stokes, K., Conley, D., & Castelle, B. (2023). Satellite-based shoreline detection along high-energy macrotidal coasts and influence of beach state. *Marine Geology*, 107082.

Krause, C., Dunn, B., Bishop-Taylor, R., Adams, C., Burton, C., Alger, M., Chua, S., Phillips, C., Newey, V., Kouzoubov, K., & others. (2021). *DEA Notebooks contributors 2021: Digital Earth Australia notebooks and tools repository, Geoscience Australia, Canberra*. https://knowledge.dea.ga.gov.au/notebooks/README/.

Lyard, F. H., Allain, D. J., Cancet, M., Carrere, L., & Picot, N. (2021). FES2014 global ocean tide atlas: Design and performance. *Ocean Science*, *17*(3), 615–649.

McLean, R., Thom, B., Shen, J., & Oliver, T. (2023). 50 years of beach–foredune change on the southeastern coast of australia: Bengello beach, moruya, NSW, 1972–2022. *Geomorphology*, *439*, 108850.

NVIDIA. (2023). *NVIDIA Model Zoo*. https://docs.nvidia.com/tao/tao-toolkit/text/model_zoo/overview.html%0A.

Planet Labs. (2018). *Planet Application Program Interface: In Space for Life on Earth*. Planet Labs. https://api.planet.com

PyTorch. (2020). *PyTorch Model Zoo*. https://pytorch.org/serve/model_zoo.html.

Scheffler, D., Hollstein, A., Diedrich, H., Segl, K., & Hostert, P. (2017). AROSICS: An automated and robust open-source image co-registration software for multi-sensor satellite data. *Remote Sensing*, *9*(7), 676.

Turner, I. L., Harley, M. D., Almar, R., & Bergsma, E. W. J. (2021). Satellite optical imagery in coastal engineering. *Coastal Engineering*, *167*, 103919.

Vandenhove, M., Castelle, B., Lerma, A. N., Marieu, V., Dalet, E., Hanquiez, V., Mazeiraud, V., Bujan, S., & Mallet, C. (2024). Secular shoreline response to large-scale estuarine shoal migration and welding. *Geomorphology*, *445*, 108972.

Vitousek, S., Buscombe, D., Vos, K., Barnard, P. L., Ritchie, A. C., & Warrick, J. A. (2023). The future of coastal monitoring through satellite remote sensing. *Cambridge Prisms: Coastal Futures*, *1*, e10.

DRAFT

Vitousek, S., Vos, K., Splinter, K. D., Erikson, L., & Barnard, P. L. (2023). A model integrating satellite-derived shoreline observations for predicting fine-scale shoreline response to waves and sea-level rise across large coastal regions. *Journal of Geophysical Research: Earth Surface*, e2022JF006936.

Vos, K. (2023). *CoastSat v2.4*. https://github.com/kvos/CoastSat/releases/tag/v2.4; GitHub.

Vos, K., & Fitzpatrick, S. (2023). Coastsat-package. In *PyPi*. https://pypi.org/project/coastsat-package/; PyPi.

Vos, K., Harley, M. D., Turner, I. L., & Splinter, K. D. (2023). Pacific shoreline erosion and accretion patterns controlled by el niño/southern oscillation. *Nature Geoscience*, *16*(2), 140–146.

Vos, K., Splinter, K. D., Harley, M. D., Simmons, J. A., & Turner, I. L. (2019). CoastSat: A Google Earth Engine-enabled Python toolkit to extract shorelines from publicly available satellite imagery. *Environmental Modelling & Software*, *122*, 104528.

Vos, K., Splinter, K. D., Palomar-Vázquez, J., Pardo-Pascual, J. E., Almonacid-Caballer, J., Cabezas-Rabadán, C., Kras, E. C., Luijendijk, A. P., Calkoen, F., Almeida, L. P., & others. (2023). Benchmarking satellite-derived shoreline mapping algorithms. *Communications Earth & Environment*, *4*(1), 345.

Warrick, J. A., Vos, K., Buscombe, D., Ritchie, A. C., & Curtis, J. A. (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.

DRAFT