

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

Sharon Fitzpatrick<sup>1</sup>, Daniel Buscombe<sup>1</sup>, Jonathan A. Warrick<sup>2</sup>, Mark A. Lundine<sup>2</sup>, and Kilian Vos<sup>3</sup>

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

DOI: [10.xxxxxx/draft](https://doi.org/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Published: unpublished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](#)).

## 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 API 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 API for specific workflows. Two fully functioning SDS workflows are already provided, and more could be added by collaborators in the SDS software community. All API 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 is an elevation-based measurement. 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 (Konstantinou et al., 2023; Vitousek, Buscombe, et al., 2023; Vitousek, Vos, et al., 2023).

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 transect-based measurements, and numerous other workflows common to a basic SDS workflow, regardless of a particular waterline estimation workflow. 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 API 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 created and maintain CoastSat-package (Vos & 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, 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 that 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 a new jupyter notebook, and also likely contribute to the CoastSeg source code to add their specific waterline detection algorithm, which could be called in their notebook.

## 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 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 within a single jupyter notebook, in a quicker, and in a more seamless manner, 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

94 shorelines using multiple methods and adopt the one that most suits their needs, or implement  
95 a new methods.

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

- 97     ▪ A re-implementation of (and improvement of) the CoastSat workflow with pip-installable  
98       APIs, and coastsat-package.
- 99     ▪ A browser-based workflow and an interactive mapping interface provided by Leafmap  
100       (Wu, 2021).
- 101     ▪ A more accessible, entirely graphical and menu-based SDS workflow, with no (mandatory)  
102       exposure of source code to the user.
- 103     ▪ A session system that streamlines the experimentation process to find the settings that  
104       extract optimal shorelines from satellite imagery.
- 105     ▪ Improved core SDS workflow components, such as a faster and more seamless tidal  
106       correction workflow, and faster image downloading.
- 107     ▪ Consolidation of workflows in a single platform and reusable codebase.
- 108     ▪ An extendable hub of alternative SDS workflows in one location.

## 109 Implementation of core SDS workflow

### 110 Architecture & Design

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

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

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

### 126 Sessions

127 SDS workflows require manipulating various settings in order to extract optimal shorelines.  
128 There are numerous settings in the CoastSat workflow, and sometimes determining optimal  
129 shorelines can be an iterative process requiring experimentation with settings. Sub-optimal  
130 shoreline extraction may result merely through user fatigue or a combination of misconfigured  
131 settings. Therefore, CoastSeg employs a session-based system that enables users to iteratively  
132 experiment with different combinations of settings. Each time the user makes adjustments to  
133 the settings used to extract shorelines from the imagery a new session folder is saved with the  
134 updated settings. This session system is what makes CoastSeg fully reproducible because all  
135 the settings, inputs, and outputs are stored within each session as well as a reference to what  
136 downloaded data was used to generate the extracted shorelines in the session. Moreover, the  
137 session system in CoastSeg fosters a collaborative environment. Users can share their sessions

138 with others, enabling peers to replicate experiments, build upon previous work, or access data  
139 downloaded by someone else. This simplifies the process for new users and encourages collective  
140 experimentation and data sharing. This reproducibility and collaboration are beneficial in  
141 research contexts.

## 142 **Improvements to the CoastSat workflow**

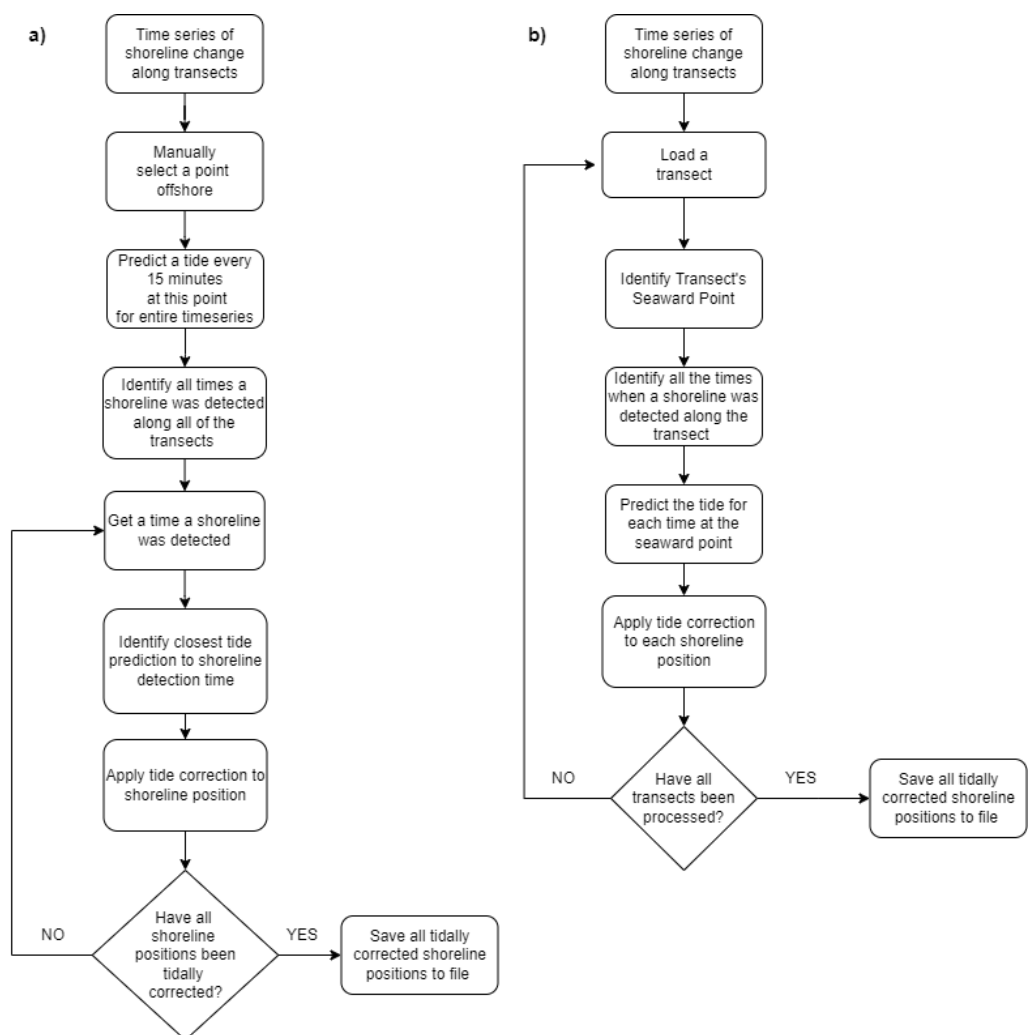
### 143 **Accessibility**

144 CoastSeg facilitates entirely browser-based workflows with an interactive webmap and  
145 ipywidget controls. It interfaces with the Zenodo API to download reference shorelines for  
146 any location in the world, organized into 5x5 degree chunks in GeoJSON format (Buscombe,  
147 2023) as well as transects, themselves providing beachface slope metadata (Buscombe &  
148 Fitzpatrick, 2023) available on hover. We have implemented rigorous error handling using  
149 developer log files, user report files, and informative error messages that suggest problem fixes.  
150 We have also provided a set of utility scripts for common data input/output tasks, often the  
151 result of specific requests from our software testers (see Acknowledgments). In addition to  
152 a project wiki and improved documentation, we have researched minimum, maximum, and  
153 recommended values for all settings, and have provided visual project management aids.

### 154 **Performance**

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

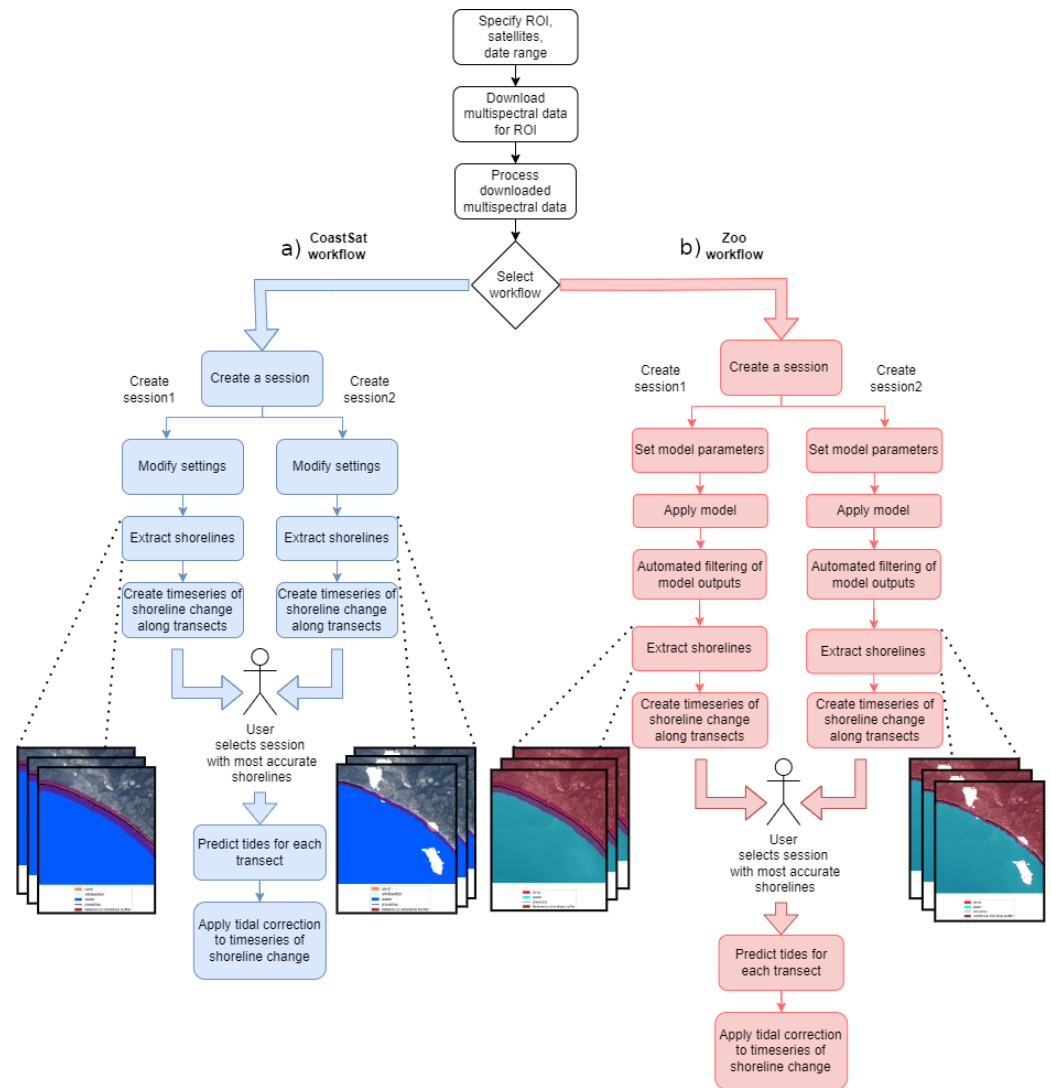
167 We added helpful workflow components such as image filtering options; for example, users can  
168 now filter their imagery based on image size and the proportion of no data pixels in an image.  
169 Additionally, the user can decide to turn off cloud masking, which is necessary when the cloud  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



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

## 174 Tide

175 Tidal correction (Figure 1) of shorelines involves estimating the tide height for any location and  
 176 time using the pyTMD API (Alley et al., 2017) to model the tide. pyTMD provides an accessible  
 177 script for the widely used FES14 (Lyard et al., 2021) tidal model data access, and includes  
 178 several models other than FES14 including polar-specific models. We created an automated  
 179 workflow that splits the FES2014 model data into 11 global regions (an idea adopted from  
 180 (Krause et al., 2021)). This allows the program to access only a subset of the data, facilitating  
 181 fast tide estimates (in minutes rather than hours for multi-decadal 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 & 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 SDS workflow adopted for waterline detection 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 API.

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 uses to estimate runup or setup from the prevailing wave field would require research and software integration.

## Acknowledgments

The author 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.

## References

- Alley, K., Brunt, K., Howard, S., Padman, L., Siegfried, M., & Sutterly, T. (2017). *PyTMD: Python based tidal prediction software*. <https://doi.org/10.5281/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.
- 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>

- 240 Buscombe, D., & Goldstein, E. (2022). A reproducible and reusable pipeline for segmentation  
241 of geoscientific imagery. *Earth and Space Science*, 9(9), e2022EA002332.
- 242 Castelle, B., Masselink, G., Scott, T., Stokes, C., Konstantinou, A., Marieu, V., & Bujan,  
243 S. (2021). Satellite-derived shoreline detection at a high-energy meso-macrotidal beach.  
244 *Geomorphology*, 383, 107707.
- 245 Castelle, B., Ritz, A., Marieu, V., Lerma, A. N., & Vandenhove, M. (2022). Primary drivers  
246 of multidecadal spatial and temporal patterns of shoreline change derived from optical  
247 satellite imagery. *Geomorphology*, 413, 108360.
- 248 Doherty, Y., Harley, M. D., Vos, K., & Splinter, K. D. (2022). A Python toolkit to monitor sandy  
249 shoreline change using high-resolution PlanetScope cubesats. *Environmental Modelling &  
250 Software*, 157, 105512.
- 251 Garcia-Rubio, G., Huntley, D., & Russell, P. (2015). Evaluating shoreline identification using  
252 optical satellite images. *Marine Geology*, 359, 96–105.
- 253 Konstantinou, A., Scott, T., Masselink, G., Stokes, K., Conley, D., & Castelle, B. (2023).  
254 Satellite-based shoreline detection along high-energy macrotidal coasts and influence of  
255 beach state. *Marine Geology*, 107082.
- 256 Krause, C., Dunn, B., Bishop-Taylor, R., Adams, C., Burton, C., Alger, M., Chua, S., Phillips,  
257 C., Newey, V., Kouzoubov, K., & others. (2021). *DEA Notebooks contributors 2021:  
258 Digital Earth Australia notebooks and tools repository*, Geoscience Australia, Canberra.  
259 <https://knowledge.dea.ga.gov.au/notebooks/README/>.
- 260 Lyard, F. H., Allain, D. J., Cancet, M., Carrere, L., & Picot, N. (2021). FES2014 global ocean  
261 tide atlas: Design and performance. *Ocean Science*, 17(3), 615–649.
- 262 McLean, R., Thom, B., Shen, J., & Oliver, T. (2023). 50 years of beach–foredune change on the  
263 southeastern coast of australia: Bengello beach, moruya, NSW, 1972–2022. *Geomorphology*,  
264 439, 108850.
- 265 NVIDIA. (2023). *NVIDIA Model Zoo*. [https://docs.nvidia.com/tao/tao-toolkit/text/model\\_  
266 zoo/overview.html%0A](https://docs.nvidia.com/tao/tao-toolkit/text/model_zoo/overview.html%0A).
- 267 Planet Labs. (2018). *Planet Application Program Interface: In Space for Life on Earth*. Planet  
268 Labs. <https://api.planet.com>
- 269 PyTorch. (2020). *PyTorch Model Zoo*. [https://pytorch.org/serve/model\\_zoo.html](https://pytorch.org/serve/model_zoo.html).
- 270 Scheffler, D., Hollstein, A., Diedrich, H., Segl, K., & Hostert, P. (2017). AROSICS: An  
271 automated and robust open-source image co-registration software for multi-sensor satellite  
272 data. *Remote Sensing*, 9(7), 676.
- 273 Turner, I. L., Harley, M. D., Almar, R., & Bergsma, E. W. J. (2021). Satellite optical imagery  
274 in coastal engineering. *Coastal Engineering*, 167, 103919.
- 275 Vandenhove, M., Castelle, B., Lerma, A. N., Marieu, V., Dalet, E., Hanquiez, V., Mazeiraud,  
276 V., Bujan, S., & Mallet, C. (2024). Secular shoreline response to large-scale estuarine shoal  
277 migration and welding. *Geomorphology*, 445, 108972.
- 278 Vitousek, S., Buscombe, D., Vos, K., Barnard, P. L., Ritchie, A. C., & Warrick, J. A. (2023).  
279 The future of coastal monitoring through satellite remote sensing. *Cambridge Prisms:  
280 Coastal Futures*, 1, e10.
- 281 Vitousek, S., Vos, K., Splinter, K. D., Erikson, L., & Barnard, P. L. (2023). A model integrating  
282 satellite-derived shoreline observations for predicting fine-scale shoreline response to waves  
283 and sea-level rise across large coastal regions. *Journal of Geophysical Research: Earth  
284 Surface*, e2022JF006936.
- 285 Vos, K. (2023). *CoastSat v2.4*. <https://github.com/kvos/CoastSat/releases/tag/v2.4>; GitHub.



- 286 Vos, K., & Fitzpatrick, S. (2023). Coastsat-package. In *PyPi*. [https://pypi.org/project/](https://pypi.org/project/coastsat-package/)  
287 [coastsat-package/](https://pypi.org/project/coastsat-package/); PyPi.
- 288 Vos, K., Harley, M. D., Turner, I. L., & Splinter, K. D. (2023). Pacific shoreline erosion and  
289 accretion patterns controlled by el niño/southern oscillation. *Nature Geoscience*, 16(2),  
290 140–146.
- 291 Vos, K., Splinter, K. D., Harley, M. D., Simmons, J. A., & Turner, I. L. (2019). CoastSat: A  
292 Google Earth Engine-enabled Python toolkit to extract shorelines from publicly available  
293 satellite imagery. *Environmental Modelling & Software*, 122, 104528.
- 294 Vos, K., Splinter, K. D., Palomar-Vázquez, J., Pardo-Pascual, J. E., Almonacid-Caballer, J.,  
295 Cabezas-Rabadán, C., Kras, E. C., Lujendijk, A. P., Calkoen, F., Almeida, L. P., & others.  
296 (2023). Benchmarking satellite-derived shoreline mapping algorithms. *Communications*  
297 *Earth & Environment*, 4(1), 345.
- 298 Warrick, J. A., Vos, K., Buscombe, D., Ritchie, A. C., & Curtis, J. A. (2023). A large sediment  
299 accretion wave along a Northern California littoral cell. *Journal of Geophysical Research:*  
300 *Earth Surface*, e2023JF007135.
- 301 Wu, Q. (2021). Leafmap: A Python package for interactive mapping and geospatial analysis  
302 with minimal coding in a Jupyter environment. *Journal of Open Source Software*, 6(63),  
303 3414.