9-24-2022

# CoastalImageLib: An open- source Python package for creating common coastal image products

Maile P. McCann

Dylan L. Anderson

Christopher R. Sherwood

Brittany Bruder

A. Spicer Bak

*See next page for additional authors*

Authors
Maile P. McCann, Dylan L. Anderson, Christopher R. Sherwood, Brittany Bruder, A. Spicer Bak, and Katherine L. Brodie

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

# CoastalImageLib: An open- source Python package for creating common coastal image products

Maile P. McCann [a,b,*], Dylan L. Anderson [b,c], Christopher R. Sherwood [d], Brittany Bruder [b], A. Spicer Bak [b], Katherine L. Brodie [b]

[a] Sonny Astani Department of Civil & Environmental Engineering, University of Southern California, Los Angeles, CA 90089, United States of America
[b] Coastal and Hydraulics Laboratory, U.S. Army Engineer Research and Development Center, 1261 Duck Rd, Kitty Hawk, NC 27949, United States of America
[c] Department of Civil, Construction, and Environmental Engineering, North Carolina State University, Raleigh, NC 27607, United States of America
[d] U.S. Geological Survey, Woods Hole Coastal and Marine Science Center, 384 Woods Hole Rd, Woods Hole, MA 02543, United States of America

ARTICLE INFO

*Article history:*
Received 2 August 2021
Received in revised form 3 September 2022
Accepted 24 September 2022

*Keywords:*
Python
Coastal imaging
Photogrammetry
Argus

ABSTRACT

*CoastalImageLib* is a Python library that produces common coastal image products intended for quantitative analysis of coastal environments. This library contains functions to georectify and merge multiple oblique camera views, produce statistical image products for a given set of images, and create subsampled pixel instruments for use in bathymetric inversion, surface current estimation, run-up calculations, and other quantitative analyses. This package intends to be an open-source broadly generalizable front end to future coastal imaging applications, ultimately expanding user accessibility to optical remote sensing of coastal environments. This package was developed and tested on data collected from the Argus Tower, a 43 m tall observation structure in Duck, North Carolina at the US Army Engineer Research and Development Center's Field Research Facility that holds six stationary cameras which collect twice-hourly coastal image products. Thus, *CoastalImageLib* also contains functions designed to interface with the file storage and collection system implemented at the Argus Tower.

## Code metadata

| | |
|---|---|
| Current code version | 1.0.0 |
| Permanent link to repository used for this version | https://github.com/ElsevierSoftwareX/SOFTX-D-21-00144 |
| Code Ocean compute capsule | Submitted for Publication. Waiting for approval. |
| Legal Code License | MIT Open Source License |
| Code versioning system used | git |
| Software code languages, tools, and services used | Python 3.10, Python 3.10 packages: *OpenCV, NumPy, SciPy, os, sys, MatPlotLib, math, imageIO* |
| Compilation requirements, operating environments & dependencies | See *CoastalImageLib* User Manual in GitHub repository |
| Link to developer documentation/manual | See *CoastalImageLib* User Manual in GitHub repository |
| Support email for questions | mailemcc@usc.edu |

## 1. Motivation and significance

Optical remote sensing has become an important tool for coastal scientists and engineers to expand research capabilities in the nearshore, providing low cost, accurate, and flexible methods for characterizing coastal environments. Optical imagery can be collected via a range of remote-sensing approaches, with typical applications using fixed cameras mounted on elevated platforms over looking the coast. The first coastal video monitoring system, called an Argus camera system (named after the figure in Greek mythology with 100 eyes), was developed by the Coastal Imaging Lab at Oregon State University in the early 1980s [1], and similar coastal imaging stations continue to be used worldwide today

* Corresponding author at: Sonny Astani Department of Civil & Environmental Engineering, University of Southern California, Los Angeles, CA 90089, United States of America.
*E-mail address:* mailemcc@usc.edu (Maile P. McCann).

[2–7]. Subsequent advances in low-cost video hardware have further broadened the use of optical remote sensing to data collected from tools such as unmanned aerial vehicle mounted cameras [8] and even free surf cameras [9].

Using this technology, algorithms quantifying nearshore dynamics from remotely-sensed products can estimate geophysical parameters such as two-dimensional surface currents [10–13], bathymetry [14–16], directional wave spectra [17,18], and shoreline position [19,20]. Additionally, recent advances in coastal imaging are beginning to leverage machine learning and computer-vision algorithms, such as optical wave gauging using deep learning [21], automating beach-state classification from trained neural networks [22], rip-current detection from neural networks [23], as well as algorithms combining synthetic and real imagery to train models for bathymetric inversion [24]. These advances motivate the need for a common coastal imaging library that can interface with all Python- based machine learning and computer vision packages such as TensorFlow [25], PyTorch [26], Scikit-learn [27], and OpenCV [28]. However many nearshore imaging platforms are either black box algorithms or have licenses preventing modification (e.g., Argus [1], Cam-Era, HORUS, CoastalCOMS, KOSTASYSTEM, COSMOS [5]). SIRENA provides a java-based code [4], while Beachkeeper plus [29] and ULISES [30] provide Matlab-based codes that can also be run in Octave to be open-source.

Additionally, all aforementioned algorithms require common photogrammetry functions during data preprocessing in order to convert oblique imagery into compatible georectified input imagery. The necessary photogrammetry functions can include georectification, which is the process of transforming pixel coordinates in oblique images into real-world coordinates, merging multiple camera views, preparing statistical image products, and/or creating selected pixel outputs from input imagery. These preprocessing steps are sometimes outside the expertise of coastal scientists, engineers, and oceanographers, which can constrain the availability of optical remote sensing algorithms to those with extensive photogrammetry backgrounds [31].

Steps have been taken to bridge the knowledge gap and teach photogrammetry fundamentals, such as the creation of Coastal Imaging Research Network (CIRN), an international network of researchers who collaborate to develop and implement coastal imaging methods [32]. CIRN researchers developed the CIRN Quantitative Imaging Toolbox [31] which serves as a user- accessible end- to- end package for creating coastal image products. However, this toolbox is built on Matlab, which can be cost- prohibitive and does not interface easily with open-source tools such as machine learning packages (TensorFlow, PyTorch, Scikit-learn) and computer vision packages like OpenCV. *CoastalImageLib* intends to be a complementary package to CIRN's Quantitative Imaging Toolbox, translating these capabilities to an open- source language. *CoastalImageLib* is adapted from capabilities in the CIRN Quantitative Coastal Imaging Library [31], the ARGUS Coastal Imaging System [1], and the CoastCam System [33] to provide an open- source package that interfaces with other useful Python based computer vision algorithms. In addition to CIRN's Quantitative Coastal Imaging Toolbox, the Picoastal camera system based off of the Argus framework was recently developed for a similar purpose [34]. However, Picoastal focuses on the hardware component as well, while the package we present in this paper will be a software package exclusively with the intention of being a build-able foundation for applications that may interface with a variety of hardware applications, not just that of Argus/CIRN/Picoastal framework.

The recent abundance of hardware and nearshore imagery holds considerable potential for coastal monitoring of both chronic and episodic hazards, motivating the need for this open-source toolbox that the coastal research community can utilize to derive inter-comparable products. Ultimately, this package aims to be a broadly generalizable, foundational front end to future coastal imaging applications that can leverage valuable computer vision and machine learning techniques.

## 2. Software description

The *CoastalImageLib* package contains modules to georectify and merge multiple camera views, calculate statistical image products, and create subsampled pixel timestacks. These products can be utilized in a wide range of optical remote sensing applications, and the library itself can interface directly with open source computer vision and machine learning algorithms such as OpenCV, PyTorch, SciPy, and TensorFlow.

### 2.1. Software architecture

The following list depicts the library structure for *CoastalImageLib*, expressed in terms of a hierarchical file system. Any classes contained in each module (*.py file) are included in italics. Classes, as well as the methods they contain, do not require a specific order in which to be implemented. Suggested workflows are included in the Illustrative Example Script Jupyter notebook contained in the *CoastalImageLib* repository. Detailed descriptions of each module can be found in the following Software Functionalities section.

**CoastalImageLib/**

- corefunctions.py

    *class XYZGrid()*
    *class CameraData()*

- supportfunctions.py
- argusIO.py

The main user- interactive module is **corefunctions.py**. Two classes are contained in this module: *XYZGrid()* and *CameraData()*. These classes bundle data and functionality vital to the rectification process. *XYZGrid()* holds the real world target grid on which rectification or pixel subsampling will take place. *CameraData()* holds camera calibration values, and contains a method for extrinsic value transforms, and a method for calculating camera matrices. Users must initialize instances of these classes for each desired rectification grid, and each calibrated camera.

### 2.2. Software functionalities: ***corefunctions.py***

#### 2.2.1. Georectification and merging multiple camera views

The module **corefunctions.py** contains a series of functions that implement fundamental photogrammetry calculations to georectify oblique imagery onto a user- defined real world XYZ grid and merge multiple camera views, for both grayscale and color images. Additionally, this module contains functions to generate statistical image products for a given set of images and their corresponding camera extrinsic and intrinsic values, as well as functions to generate pixel instruments for use in bathymetric inversion, surface current, or run-up calculations. For rectification tasks, the user first initializes an *XYZGrid()* object. The user specifies $x$ and $y$ limits and resolution of the real- world grid in $x$ and $y$ directions. See the *CoastalImageLib* User Manual for how coordinate systems are defined.

Next, the user initializes a *CameraData()* object for each calibrated camera being utilized. Each instance of this class requires

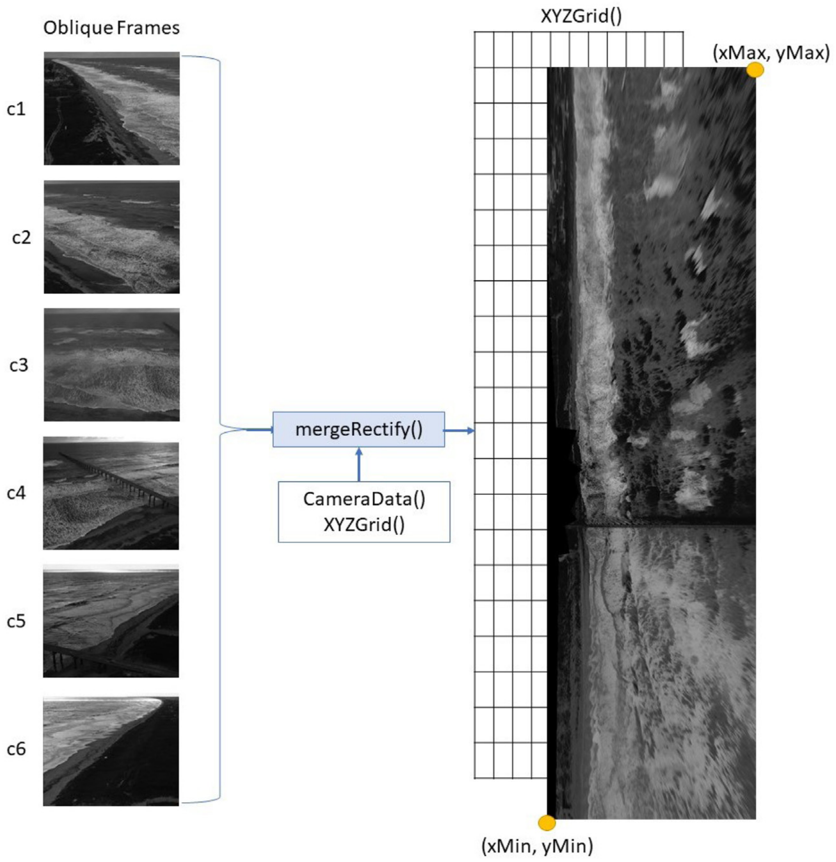

**Fig. 1.** Diagram of the rectification process, shown in grayscale. However, both grayscale and color images are supported.

all camera intrinsic and extrinsic values unique to that device. For cameras that have not yet been calibrated and the intrinsic values are not known, the user is directed to the CalTech Camera Calibration library [35], or other relevant calibration libraries such as the calibration functions contained on OpenCV [28]. Intrinsic values are accepted in the direct linear transform notation [36] which has been adopted by CIRN as common convention [31]. See the *CoastalImageLib* User Manual for detailed information on calibration and intrinsic value formatting. The user can also optionally specify the coordinate system being utilized, with the further option of providing the local origin for a coordinate transform.

If oblique imagery was captured using a non-stationary camera, for example an unmanned aerial vehicle mounted camera, the user is directed to the CIRN Quantitative Coastal Imaging library for calibration and stabilization [31]. Note that this library requires stationary ground control points (GCPs) and stabilization control points (SCPs). See the CIRN Quantitative Coastal Imaging library User Manual [31] for detailed information on GCPs and SCPs.

The **corefunctions.py** function *mergeRectify()* is designed to merge and rectify one or more cameras at one timestamp into a single frame, as shown in Fig. 1. For multiple subsequent frames, the user can either loop through *mergeRectify()* and rectify each desired frame on the same XYZ grid, or call the function *rectVideos()* to merge and rectify frames from one or more cameras provided in video format, sampled at the same time and frame rate. Merging of multiple cameras includes a histogram matching step, where the histogram of the first camera view is used as the reference histogram for balancing subsequent camera views. This step helps remove visible camera seams and improves the congruity of illumination [28].

### 2.2.2. *Statistical image products*

The **corefunctions.py** module also contains the function *imageStats()* to generate statistical image products for a given set of stationary oblique or georectified images contained in a three dimensional array, in either grayscale or color. All image product calculations are taken from the Argus video monitoring convention [1]. The products and their descriptions are as follows:

1. Brightest: These images are the composite of all the brightest (maximum) pixel intensities at each pixel location throughout the entire collection.
2. Darkest: These images are the composite of all the darkest (minimum) pixel intensities at each pixel location throughout the entire collection. In regions of intermittent breaking, Darkest images have historically been used to look through the water column [37].
3. Timex: Time- exposure (timex) images represent the mathematical time- mean of all the frames captured over the period of sampling. Moving features, including waves and vessels, are averaged out and only mean brightness is returned. Areas of repeated wave breaking in the surf zone appear as white bands, which can help locate and determine the morphology of sand bars and rip channels [15].
4. Variance: Variance images are found from the variance of image intensities of all the frames captured over the period of sampling. Variance images are the brightest where they have the most variation. Variance images are primarily used to delineate the surf zone and regions of wave breaking [1].

### 2.2.3. *Pixel products*

The **corefunctions.py** module also contains the function **pixelStack** to create subsampled pixel timestacks in either grayscale or

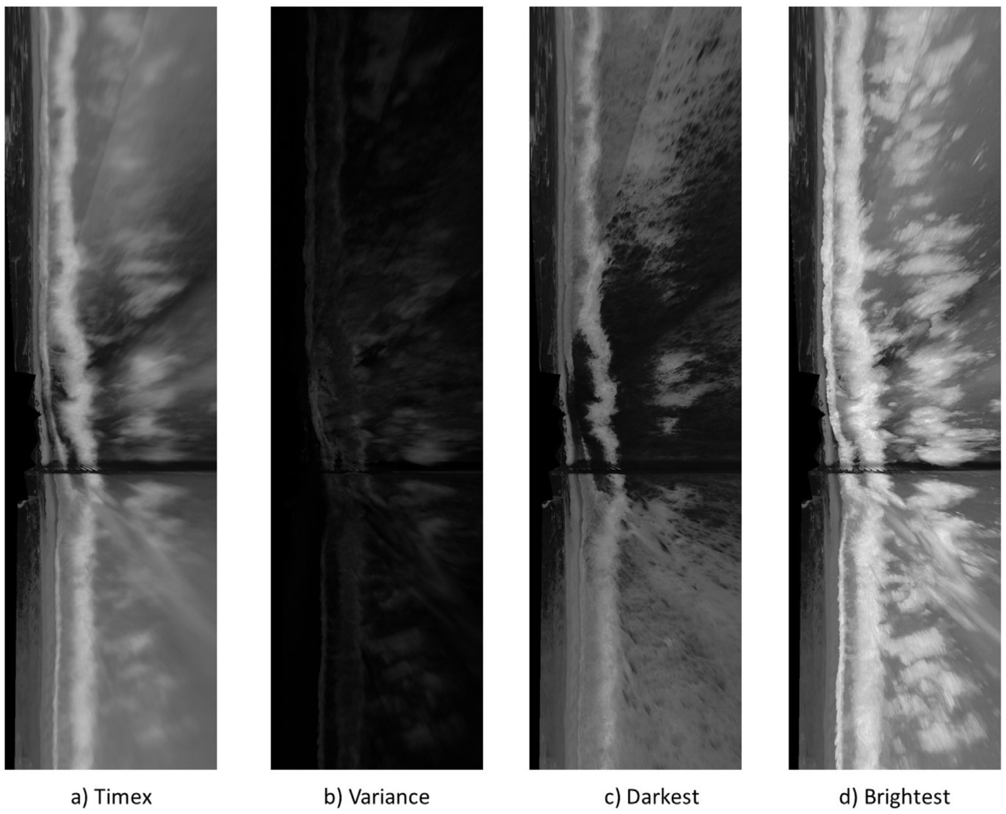

**Fig. 2.** Examples of each of the statistical image products calculated by **imageStats**.

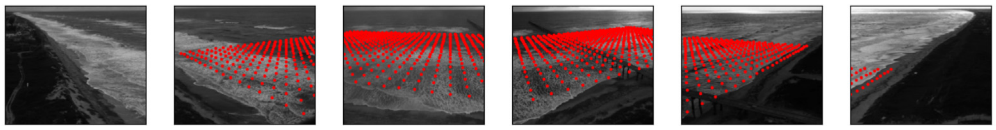

**Fig. 3.** Pixel locations plotted on input oblique images from each of the six Argus cameras.

color for use in algorithms such as bathymetric inversion, surface current estimation, or run-up calculations. Pixel timestacks show variations in pixel intensity over time. The main pixel products are included below, however additional instruments can be created from these main classes. For example, a single pixel, which may be useful for estimating wave period [38], can be generated by creating an alongshore transect of length 1.

1. Grid (also known as Bathy Array in Holman and Stanley 2007 and other publications that reference Argus image products [1]): This is a 2D array of pixels covering the entire nearshore, which can be utilized in bathymetry estimation algorithms [14]. Example grid products are shown in Figs. 3 and 4.
2. Alongshore/ Y Transect (sometimes referred to as Vbar [1]): This product is commonly utilized in estimating longshore currents [39].
3. Cross- shore/ X Transect (sometimes referred to as Runup Array [1,38]): Cross- shore transects can be utilized in estimating wave runup. Alongshore and cross- shore pixel instruments are depicted in Fig. 5.

### 2.3. *Software functionalities:* ***supportfunctions.py***

This module contains functions independent of any overarching class or specific workflow, which serve to assist the user in utilizing the core functions. **supportfunctions.py** contains supporting functions to format intrinsic files, convert extrinsic coordinates to and from geographical and local coordinate systems, calculate extrinsic values, and other steps necessary to utilize the core functions of the *CoastalImageLib* library. Additionally, **supportfunctions.py** contains functions that interface with Argus technology, camera systems initially developed by the Coastal Imaging Lab at Oregon State University [1], including functions to create Argus compatible filenames from UTC timestamps, and convert raw (*.raw*) Argus files into delivery files collected from the Argus camera systems. Converting raw Argus data utilizes functions contained in **argusIO.py**. However **argusIO.py** will not be further discussed in this paper because it does not apply to data collected outside of the Argus system. The **argusIO.py** module is included in the library for ease of use for Argus specific applications. See the *CoastalImageLib* User Manual for more detailed documentation of **supportfunctions.py**, and further discussion of **argusIO.py**.

## 3. Illustrative example: Fixed multi camera demo data

For an interactive example script working through the example data, users are directed to the Jupyter Notebook file contained in the *CoastalImageLib* repository entitled *CoastalImageLib _Illustrative_Example.ipynb*. This script walks the user through five main functionalities of the *CoastalImageLib* toolbox:

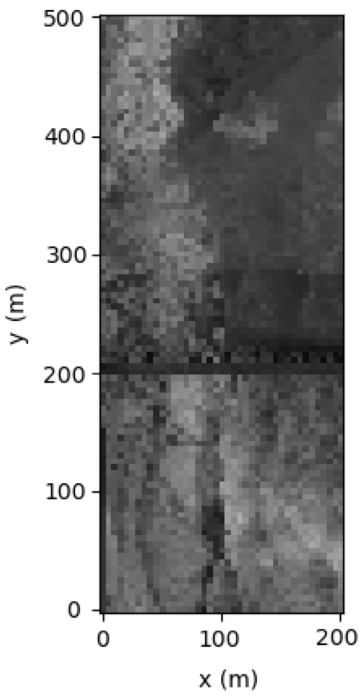

**Fig. 4.** Output pixel grid at the pixel locations shown in Fig. 3, with a resolution of 5 m.

1. *Rectifying single oblique frames from multiple cameras*. Oblique inputs as well as the correct georectified output from the example script are shown in Fig. 1.
2. *Rectifying multiple oblique frames from multiple cameras*
3. *Rectifying oblique videos from multiple cameras*
4. *Creating a pixel timestack*. The intended output from the pixel grid created in this section is shown in Fig. 4, and the pixel locations projected onto the oblique input frames are shown in Fig. 3. If the user looped through a longer dataset of images and retrieved subsampled pixels at alongshore and cross- shore transects from multiple subsequent images, the result should look like the plots displayed in Fig. 5.
5. *Calculating image statistics*. The intended output from provided example data is included in Fig. 2.

The provided Fixed Multi-Camera Demo Data is from six fixed cameras on top of the Argus Tower [1] in Duck, NC. Data was subsampled from oblique videos that were initially 17 min long and captured at 2 frames per second. Each video was recorded simultaneously. Extrinsic and intrinsic values for each camera are provided in both direct linear transform coefficient format as well as in CIRN convention.

## 4. Impact

An intended impact of *CoastalImageLib*, consistent with the CIRN ideology [31], is to reduce barriers of entry to photogrammetry that coastal engineers, geoscientists, and oceanographers may face when exploring optical remote sensing. Additionally, this package serves to be a broadly generalizable building block on which the community can expand. The recent abundance of low-cost camera hardware and available nearshore imagery holds considerable potential for future coastal monitoring of both chronic and episodic hazards, motivating the need for this open-source toolbox that the entire community can build on to derive inter-comparable products. Ultimately, this library aims to increase quantitative coastal studies from optical remote sensing in expanded locations, environmental conditions, and spatial or temporal scales.

Unlike most previous optical remote sensing packages for coastal environments, *CoastalImageLib* is open-source, as Python and the additional required packages are free and publicly available, therefore providing a foundational package on which to build more open source coastal imaging packages. Since this library utilizes Python, future builds on this library can exploit valuable open-source Python packages for computer vision and machine learning, such as OpenCV, TensorFlow, PyTorch, and SciPy. The ability to harness these capabilities opens the door for extensive development in the way of coastal imaging, utilizing new techniques that have previously gone unexplored in optical remote sensing of coastal environments.

## 5. Conclusions

*CoastalImageLib* is a Python- based library that produces georectified images as well as common coastal image products intended for quantitative analysis of coastal environments. This library contains functions to georectify and merge multiple oblique camera views, produce statistical image products for a given set of images, as well as create subsampled pixel instruments for use in bathymetric inversion, surface current, run-up calculations, and other quantitative analyses. Additionally, this library contains support functions to format camera intrinsic values from various input file formats, convert extrinsic values from geographical to user defined local coordinates, and functions to interface with Argus-based camera systems. This package intends to be an open- source broadly generalizable front end to future coastal imaging applications, ultimately expanding user accessibility to optical remote sensing of coastal environments.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

The link to data and code is included in the paper.

## Acknowledgments

This work is funded by the USACE Civil Works Coastal And Ocean Data Systems Program, the Remote Sensing Coastal Change and the Total Water Level/ Coastal Change projects under the USGS Coastal Hazards and Resources Program. Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government. This material is based upon work supported in part by the National Science Foundation, United States of America under Grants CMMI-1661052 and ICER-1940351. Any opinions,findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. The authors would like to thank Rob Holman, Kent Hathaway, and John Stanley for their help in collecting the data and paving the way with Argus.

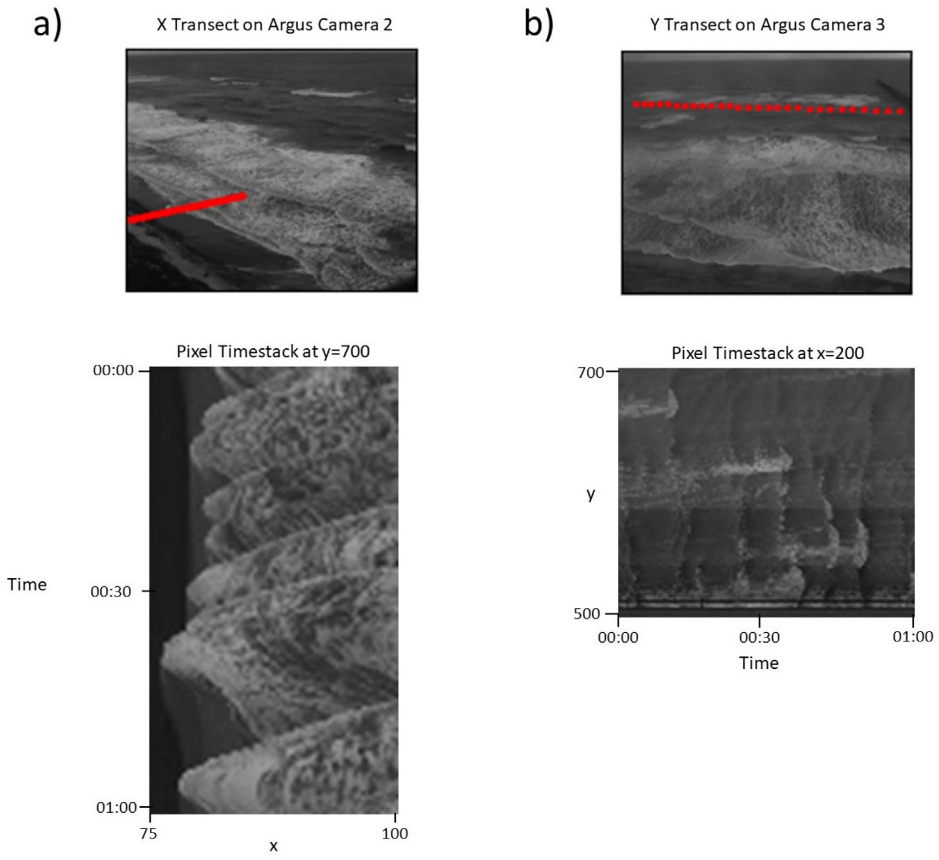

**Fig. 5.** (a) Pixel locations of an $x$ transect shown on an oblique image taken from Argus camera 2, and the pixel timestack taken from that transect over the course of 1 min, (b) Pixel locations of a $y$ transect shown on an oblique image taken from Argus camera 3, and the pixel timestack taken from that transect over the course of 1 minute.

## References

[1] Holman R, Stanley J. The history and technical capabilities of Argus. Coast Eng 2007;54(6):477–91. http://dx.doi.org/10.1016/j.coastaleng.2007.01.003, The CoastView Project: Developing coastal video monitoring systems in support of coastal zone management.

[2] Splinter KD, Harley MD, Turner IL. Remote sensing is changing our view of the coast: Insights from 40 years of monitoring at Narrabeen-collaroy, Australia. Remote Sens 2018;10(11). http://dx.doi.org/10.3390/rs10111744, URL https://www.mdpi.com/2072-4292/10/11/1744.

[3] Davidson MS, van Koningsveld M, de Kruif A, Rawson J, Holman RA, Lamberti A, et al. The CoastView project: Developing video-derived coastal state indicators in support of coastal zone management. Coast Eng 2007;54:463–75.

[4] Nieto MA, Garau B, Balle S, Simarro G, Zarruk GA, Ortiz A, et al. An open source, low cost video-based coastal monitoring system. 2010, http://dx.doi.org/10.1002/esp.2025.

[5] Taborda R, Silva A. COSMOS: A Lightweight Coastal video monitoring system. Comput Geosci 2012;49:248–55. http://dx.doi.org/10.1016/j.cageo.2012.07.013.

[6] Valentini N, Saponieri A, Damiani L. A new video monitoring system in support of coastal zone management at apulia region, Italy. Ocean Coast Manag 2017;142:122–35. http://dx.doi.org/10.1016/j.ocecoaman.2017.03.032, URL https://www.sciencedirect.com/science/article/pii/S0964569117303253.

[7] Power HE, Kinsela MA, Stringari CE, Kendall MJ, Morris BD, Hanslow DJ. Automated sensing of wave inundation across a rocky shore platform using a low-cost camera system. Remote Sens 2018;10(1). http://dx.doi.org/10.3390/rs10010011, URL https://www.mdpi.com/2072-4292/10/1/11.

[8] Holman RA, Brodie KL, Spore NJ. Surf zone characterization using a small quadcopter: Technical issues and procedures. IEEE Trans Geosci Remote Sens 2017;55(4):2017–27. http://dx.doi.org/10.1109/TGRS.2016.2635120.

[9] Conlin MP, Adams PN, Wilkinson B, Dusek G, Palmsten ML, Brown JA. SurfRCaT: A tool for remote calibration of pre-existing coastal cameras to enable their use as quantitative coastal monitoring tools. SoftwareX 2020;12:100584. http://dx.doi.org/10.1016/j.softx.2020.100584, URL https://www.sciencedirect.com/science/article/pii/S2352711020302971.

[10] Anderson D, Bak AS, Brodie KL, Cohn N, Holman RA, Stanley J. Quantifying optically derived two-dimensional wave-averaged currents in the surf zone. Remote Sens 2021;13(4). http://dx.doi.org/10.3390/rs13040690.

[11] Holman R, Haller MC. Remote sensing of the nearshore. Annu Rev Mar Sci 2013;5(1):95–113. http://dx.doi.org/10.1146/annurev-marine-121211-172408.

[12] Haller M, Honegger D, Catalán P. Rip current observations via marine radar. J Waterw Port Coast Ocean Eng 2014;140:115–24. http://dx.doi.org/10.1061/(ASCE)WW.1943-5460.0000229.

[13] Shen C, Huang W, Gill EW, Carrasco R, Horstmann J. An algorithm for surface current retrieval from X-band marine radar images. Remote Sens 2015;7(6):7753–67. http://dx.doi.org/10.3390/rs70607753.

[14] Holman R, Plant N, Holland T. cBathy: A robust algorithm for estimating nearshore bathymetry. J Geophys Res Oceans 2013;118(5):2595–609. http://dx.doi.org/10.1002/jgrc.20199.

[15] Lippmann TC, Holman RA. Quantification of sand bar morphology: A video technique based on wave dissipation. J Geophys Res Oceans 1989;94(C1):995–1011. http://dx.doi.org/10.1029/JC094iC01p00995.

[16] Dérian P, Almar R. Wavelet-based optical flow estimation of instant surface currents from shore-based and UAV videos. IEEE Trans Geosci Remote Sens 2017;50:5790–7. http://dx.doi.org/10.1109/TGRS.2017.2714202.

[17] Plant NG, Holland KT, Haller MC. Ocean wavenumber estimation from wave-resolving time series imagery. IEEE Trans Geosci Remote Sens 2008;46(9):2644–58. http://dx.doi.org/10.1109/TGRS.2008.919821.

[18] Harley M, Turner I, Short A, Ranasinghe R. Assessment and integration of conventional, RTK-GPS and image-derived beach survey methods for daily to decadal coastal monitoring. Coast Eng 2011;58:194–205. http://dx.doi.org/10.1016/j.coastaleng.2010.09.006.

[19] Plant NG, Aarninkhof SGJ, Turner IL, Kingston KS. The performance of shoreline detection models applied to video imagery. J Coast Res 2007;23(3):658–70.

[20] Alexander PS, Holman RA. Quantification of nearshore morphology based on video imaging. Mar Geol 2004;208(1):101–11. http://dx.doi.org/10.1016/j.margeo.2004.04.017.

[21] Buscombe D, Carini RJ, Harrison SR, Chickadel CC, Warrick JA. Optical wave gauging using deep neural networks. Coast Eng 2020;155:103593.

[22] Ellenson AN, Simmons JA, Wilson GW, Hesser TJ, Splinter KD. Beach state recognition using Argus imagery and convolutional neural networks. Remote Sens 2020;12(23). http://dx.doi.org/10.3390/rs12233953.

[23] de Silva A, Mori I, Dusek G, Davis J, Pang A. Automated rip current detection with region based convolutional neural networks. 2021, CoRR abs/2102.02902.

[24] Collins AM, Brodie KL, Bak AS, Hesser TJ, Farthing MW, Lee J, et al. Bathymetric inversion and uncertainty estimation from synthetic surf-zone imagery with machine learning. Remote Sens 2020;12(20). http://dx.doi.org/10.3390/rs12203364.

[25] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. 2016, arXiv preprint arXiv:1603.04467.

[26] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: An imperative style, high-performance deep learning library. Adv Neural Inf Process Syst 2019;32.

[27] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.

[28] Bradski G. The OpenCV library. Dobb's J Softw Tools 2000;120:122–5.

[29] Brignone M, Schiaffino C, Isla F, Ferrari M. A system for beach video-monitoring: Beachkeeper plus. Comput Geosci 2012;49:53–61. http://dx.doi.org/10.1016/j.cageo.2012.06.008.

[30] Simarro G, Ribas F, Alvarez A, Guillen J, Chic O, Orfila A. ULISES: An open source code for extrinsic calibrations and planview generations in coastal video monitoring systems. J Coast Res 2017;33(5):1217–27. http://dx.doi.org/10.2112/JCOASTRES-D-16-00022.

[31] Bruder BL, Brodie KL. CIRN Quantitative Coastal imaging toolbox. SoftwareX 2020;12:100582. http://dx.doi.org/10.1016/j.softx.2020.100582.

[32] Palmsten M, Brodie K. The Coastal Imaging Research Network (CIRN). Remote Sens 2022;14:453. http://dx.doi.org/10.3390/rs14030453.

[33] Sherwood C. Using video imagery to study head of the meadow beach. 2021, URL https://www.usgs.gov/centers/whcmsc/science/using-video-imagery-study-head-meadow-beach.

[34] Picoastal: A low-cost coastal video monitoring system. SoftwareX 2022;18:101073. http://dx.doi.org/10.1016/j.softx.2022.101073, URL https://www.sciencedirect.com/science/article/pii/S2352711022000541.

[35] Bouguet J-Y. Camera calibration library for Matlab, 1080. 2008, http://www.vision.caltech.edu/bouguetj/calib_doc.

[36] Abdel-Aziz YI, Karara HM. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. Photogramm Eng Remote Sens 1971;81:103–7.

[37] Clarke LB, Werner BT. Synoptic imaging of nearshore bathymetric patterns. J Geophys Res 2003;108:5–1–5–13.

[38] Stockdon HF, Holman RA. Estimation of wave phase speed and nearshore bathymetry from video imagery. J Geophys Res Oceans 2000;105(C9):22015–33. http://dx.doi.org/10.1029/1999JC000124.

[39] Chickadel C, Holman R, Freilich M. An optical technique for the measurement of longshore currents. J Geophys Res 2003;108. http://dx.doi.org/10.1029/2003JC001774.