SoftwareX 12 (2020) 100584

Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

SurfRCaT: A tool for remote calibration of pre-existing coastal cameras to enable their use as quantitative coastal monitoring tools

Matthew P. Conlin [a,*], Peter N. Adams [a], Benjamin Wilkinson [b], Gregory Dusek [c], Margaret L. Palmsten [d], Jenna A. Brown [d]

[a] Department of Geological Sciences, University of Florida, Gainesville, FL 32611, USA
[b] Department of Geomatics, University of Florida, Gainesville, FL 32611, USA
[c] Center for Operational Oceanographic Products and Services, National Ocean Service, National Oceanic and Atmospheric Administration, Silver Spring, MD 20910, USA
[d] U.S. Geological Survey, St. Petersburg Coastal and Marine Science Center, St. Petersburg,, FL 33701, USA

ARTICLE INFO

Article history:
Received 26 May 2020
Received in revised form 18 August 2020
Accepted 27 August 2020

Keywords:
Coastal monitoring
Surf-cameras
Camera calibration
Photogrammetry
Lidar

ABSTRACT

The Surf-camera Remote Calibration Tool (SurfRCaT) is a Python-based software application to calibrate and rectify images from pre-existing video cameras that are operating at coastal sites in the United States. The software enables remote camera calibration and subsequent image rectification by facilitating the remote-extraction of ground control points using airborne lidar observations, and guides the user through the entire process. No programming or code interaction are necessary to use the software. Calibration parameters and subsequent rectified image products derived from the software are saved locally. Users can apply SurfRCaT to any camera imagery that has stationary structures within the camera's field of view. Given current recreational camera infrastructure, SurfRCaT could increase the number of potential quantitative coastal video monitoring stations in the United States by an order of magnitude.

Code metadata

Current code version	*v1.1*
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_228
Code Ocean compute capsule	*NA*
Legal Code License	*GPL-3.0*
Code versioning system used	*git*
Software code languages, tools, and services used	*Python 3.6, PyQt, fbs*
Compilation requirements, operating environments & dependencies	*Anaconda Python 3 installation with pyqt, pptk, fbs, open-cv, requests, matplotlib, numpy, ftplib, pandas, reverse_geocoder, pyshp, pdal, utm, lxml*
If available Link to developer documentation/manual	https://conlin-matt.github.io/SurfRCaT/
Support email for questions	conlinm@ufl.edu

Software metadata

Current software version	*v1.1*
Permanent link to executables of this version	https://github.com/conlin-matt/SurfRCaT/releases
Legal Software License	*GPL-3.0*
Computing platforms/Operating Systems	*Microsoft Windows*
Installation requirements & dependencies	*Windows 10*
If available, link to user manual–if formally published include a reference to the publication in the reference list	https://conlin-matt.github.io/SurfRCaT/
Support email for questions	conlinm@ufl.edu

1. Motivation and significance

Beaches are dynamic landscapes shaped by oceanographic, atmospheric, and geologic processes [1–3]. Simultaneously, they

* Correspondence to: 241 Williamson Hall, University of Florida, Gainesville, FL 32611, United States of America.
E-mail address: conlinm@ufl.edu (M.P. Conlin).

https://doi.org/10.1016/j.softx.2020.100584

are among the most populated locations on Earth, with most of the world's mega-cities located in low-lying coastal regions [4]. This dichotomy creates hazards to life and property for much of the global population. Understanding the geophysical processes operating on beaches will improve our capability to predict and mitigate these hazards.

Suitable datasets of coastal geophysical processes are a fundamental prerequisite to this understanding. A variety of methods to obtain such datasets have been developed, among them the use of long-term video monitoring. Coastal researchers began deploying video camera stations on beaches in the 1980s, led by the Argus program developed at Oregon State University [5]. Simultaneously, photogrammetric and computer vision techniques were developed and applied to extract geophysical information from the optical signatures in video-image observations [6]. Application of coastal imaging systems has improved our understanding of wave runup dynamics [7], subtidal sandbar migration dynamics [8], and rip current dynamics [9], among many others. Today, over 30 Argus imaging stations are actively operating on the coasts of eight countries, in addition to other video-imaging research programs that have emerged in recent years [10–12].

Research-grade video-imaging stations are, however, confined to a relatively small number of sites, limiting those at which high-resolution video-imaging studies can be completed. In contrast, many non-research surf-cameras (surfcams; near real-time web-streaming video cameras installed on or near a beach) are already operational along coastlines around the world. In the United States (U.S.), surfcams are widespread; analysis revealed at least 328 surfcams currently operating within the country [13]. Leveraging surfcams for research purposes, as others have begun to explore only recently [12,14–16], would therefore increase the number of coastal video monitoring systems in the U.S. by an order of magnitude. This is the purpose of the software described herein.

The Surf-camera Remote Calibration Tool (SurfRCaT) builds upon the advances of previous work by providing an open-source, graphical-user-interface (GUI) driven method to remotely calibrate and rectify images from surfcams in the U.S. Camera calibration here refers to the process of determining the intrinsic and extrinsic parameters of a camera such that its images can be used to extract metric information of the imaged scene (i.e. distances), while rectification refers to the transformation of an image to a planimetric map [17]. Our Python-based tool combines a fully remote and self-contained calibration and rectification methodology with an application front-end, making it unique from other developed techniques to calibrate coastal cameras [10–12,18–20]. The application front-end of the software is downloadable and executable without any interaction with the back-end code, making it usable by groups other than coastal researchers who may not be comfortable with and/or have access to programming resources. Theoretically, SurfRCaT can be applied to any coastal camera that images hard structures (e.g. buildings, life-guard towers, signs) identifiable in airborne lidar observations.

SurfRCaT also capitalizes on a network of surfcams in the southeastern U.S. deployed by the National Oceanic and Atmospheric Administration (NOAA) National Ocean Service (NOS) and the Southeast Coastal Ocean Observing Regional Association (SECOORA) known as the Web Camera Application Testbed (WebCAT) [21]. The tool facilitates calibration of these cameras using built-in options, enabling researchers to utilize them to further ongoing research campaigns such as validation of rip-current forecast models [21]. Beyond this, SurfRCaT could transform many previously installed surfcams into quantitative coastal monitoring devices, potentially expanding the number of coastal sites around the U.S. available for research observations.

2. Software description

SurfRCaT is distributed as an application (with installer), and can be downloaded with one click or built from source code. The GUI front-end of the tool guides the user through all stages of the remote camera calibration and rectification processes. To perform these remotely, SurfRCaT facilitates the co-location of ground control points (GCPs) in both an image from the camera and a point cloud of airborne lidar observations within the imaged scene, which, unlike other sources of wide-spread coastal data such as satellite imagery [2], include elevation information. Fig. 1 shows a graphical depiction of the software architecture and functionality, described in detail below.

2.1. Software architecture

The source code for the tool is primarily contained within two *Python* files, *main.py* and *SurfRCaT.py*. The *main.py* file creates and controls the GUI front-end, while *SurfRCaT.py* contains functions to complete the calibration and rectification and is periodically called by *main.py* via standalone threads. The tool is run as a series of windows, each of which is its own *Python* class. The GUI is created using *PyQt* [22]. If built from source, the GUI is run using *fbs* [23].

The calibration and rectification processes are completed in five steps (colored boxes in Fig. 1). SurfRCaT also includes a utility to download WebCAT imagery (top colored box, Fig. 1). Below, the five steps and extra utility are described.

Inputs and imagery step — Here the user is asked to choose a working directory. If calibrating a non-WebCAT camera, the user is asked to provide estimates of the camera's location, elevation, and azimuth viewing angle, and a locally saved video from the camera (*.mp4* format; the documentation provides guidance for obtaining videos from surfcams). If calibrating a WebCAT camera, the user is assumed to have downloaded a video clip from a WebCAT camera using the Download WebCAT Utility (see below), and all other input parameters are pre-loaded. A directory is created in the working directory named *calibration_video_name*, and in it the following four subdirectories are made: *_binaries, frames, products, results*. The user is then prompted to input the rate at which they would like to save frames from the video (e.g. 1 frame per second or *n* total frames). The user can then scroll through all extracted frames and choose the one to use in the remote-GCP extraction.

Lidar data step — If a non-WebCAT camera is being calibrated, SurfRCaT interfaces with FTP servers hosted by NOAA to parse through thousands of airborne lidar datasets [24]. The U.S. state and coast are derived from the input location of the camera, and datasets are first sifted by these keywords. Remaining datasets are then examined to determine their spatial coverage by querying a *.csv* file present in each describing its geographic extent. The IDs of all datasets that have a spatial extent that includes the input location of the camera are kept. For WebCAT cameras, applicable lidar IDs come pre-loaded (Fig. 1). Applicable datasets are shown to the user and a user-chosen dataset is downloaded via the *Python* extension for the *Point Data Abstraction Library* (*pdal*) [25]. A region of interest stretching 500 m in the estimated azimuth of the camera's view (optical axis), plus or minus a 20 degree tolerance, is defined, and lidar tiles that intersect this region are downloaded. Downloaded data are formatted as a three column (X, Y, Z) point cloud and saved locally in the *products* subdirectory in a compressed file format (*.pkl*; Fig. 1).

GCPs step — The saved lidar point cloud is then displayed alongside the image to facilitate the co-location of points visible in both datasets (GCPs). The lidar point cloud, which typically

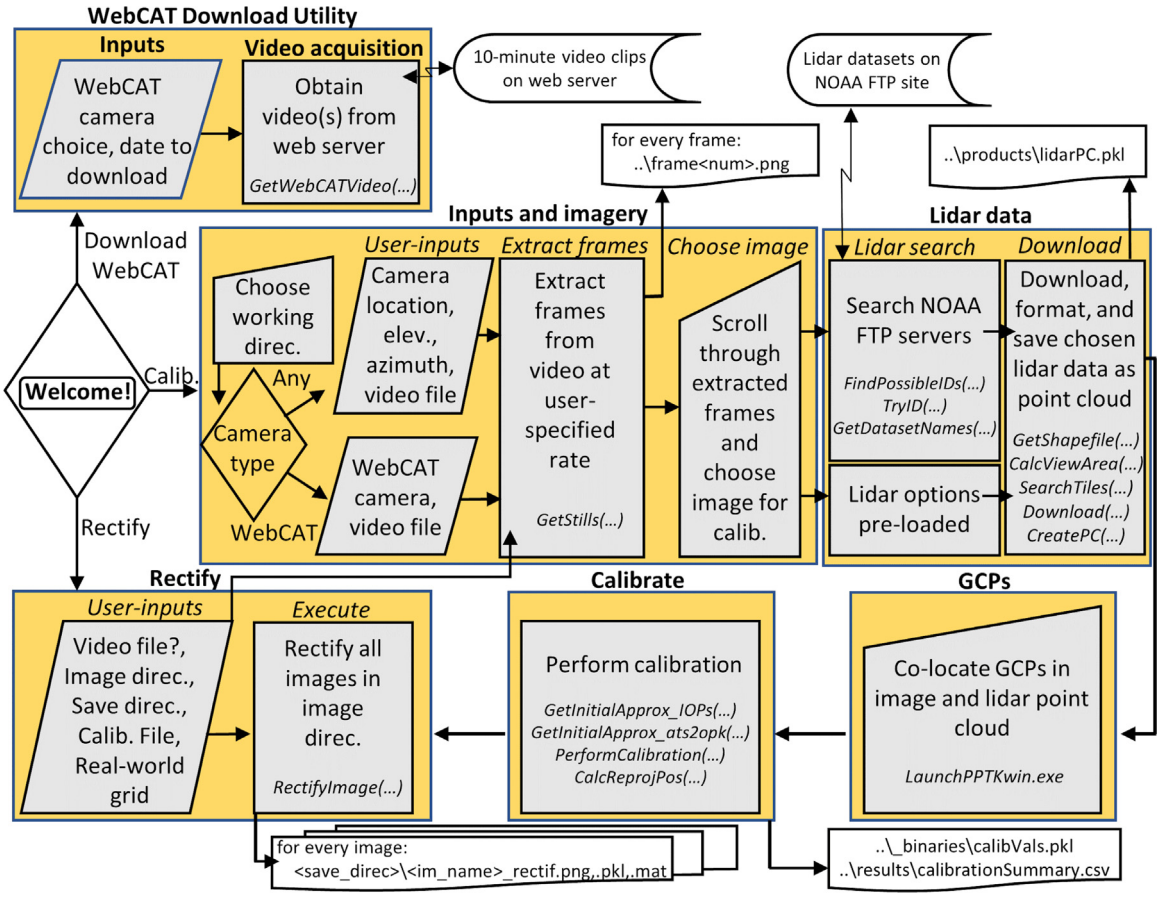

Fig. 1. SurfRCaT functionality and architecture flow-chart. Each of the five steps and extra utility are shown with a colored box.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

contains a million points or more, is displayed using the *Point Processing Toolkit* (*pptk*) [26]. A call to the *pptk* package invokes a point cloud viewer that opens as a stand-alone operating system process. Points can be selected in the viewer and are automatically output as *Python* variables. By prompting the user to select corresponding points in the image and lidar dataset, SurfRCaT saves locally both the image coordinates and real-world geographic coordinates of the selected GCPs. To interface with the viewer from the application, a *Python* script calling the viewer and saving its output locally was created and frozen to a standalone executable using *Pyinstaller* [27]. This executable is invoked from the SurfRCaT app as a separate process from the main app event-loop, such that it is not blocked by the main app. Selected points output from the viewer executable are loaded into the main app upon closure of the viewer window.

Calibrate step — Remotely-extracted GCPs are then used to calibrate the camera following the methods presented in [17]. The nine parameters included in the calibration are: camera location (x and y) and elevation (z), three camera view angles, camera focal length, and camera principal point coordinates (x and y). Initial approximations for all parameters are necessary to complete the calibration and are obtained via user inputs (for camera location, elevation, and azimuth) and software assumptions. The two other camera look-angles, tilt and roll, are estimated at 80° and 180°, respectively. An initial approximation for camera focal length is derived by using the image width and assuming a horizontal field of view of 60°. Finally, an initial approximation for the principal point location is taken as the center of the input image. SurfRCaT does not account for lens distortion in its calibration. SurfRCaT may therefore not be applicable to surfcams that exhibit large distortion characteristics or with parameters that deviate substantially from the initial parameter assumptions described above.

This step results in a binary file containing the values of the nine camera parameters, saved to the *_binaries* subdirectory as *calibVals.pkl*. This step also produces a file named *calibrationSummary.csv* in the *results* subdirectory which provides information about the calibration process in a human-readable format.

Rectify step — Rectified planimetric maps are then created using saved calibration parameter values. Users can skip directly to this section from the first window of the tool. Otherwise, this step follows the Calibrate step (Fig. 1). The user is first able to interface with the frame extraction section of the Inputs and imagery step to extract frames from a video and save them to any directory. The user then specifies a directory containing the image(s) to be rectified. To perform the rectification, the user specifies a grid in real-world space, relative to the camera location, onto which the images are projected by applying the optimized calibration parameters within the *calibVals.pkl* file. The rectification takes place planimetrically at a user-input elevation, and each image to be rectified must be assigned its own rectification elevation. Rectified image products are saved to a user-specified directory alongside a compressed *Python* dictionary and *Matlab* structure that contain the image data.

WebCAT download utility- SurfRCaT also provides a utility to download videos from WebCAT cameras. This utility can be accessed from the first window of the tool (Fig. 1). WebCAT videos are stored in 10-min clips on web servers hosted by Axiom Data Science, made openly available in partnership with SECOORA [21]. The user is asked to choose a WebCAT camera and the

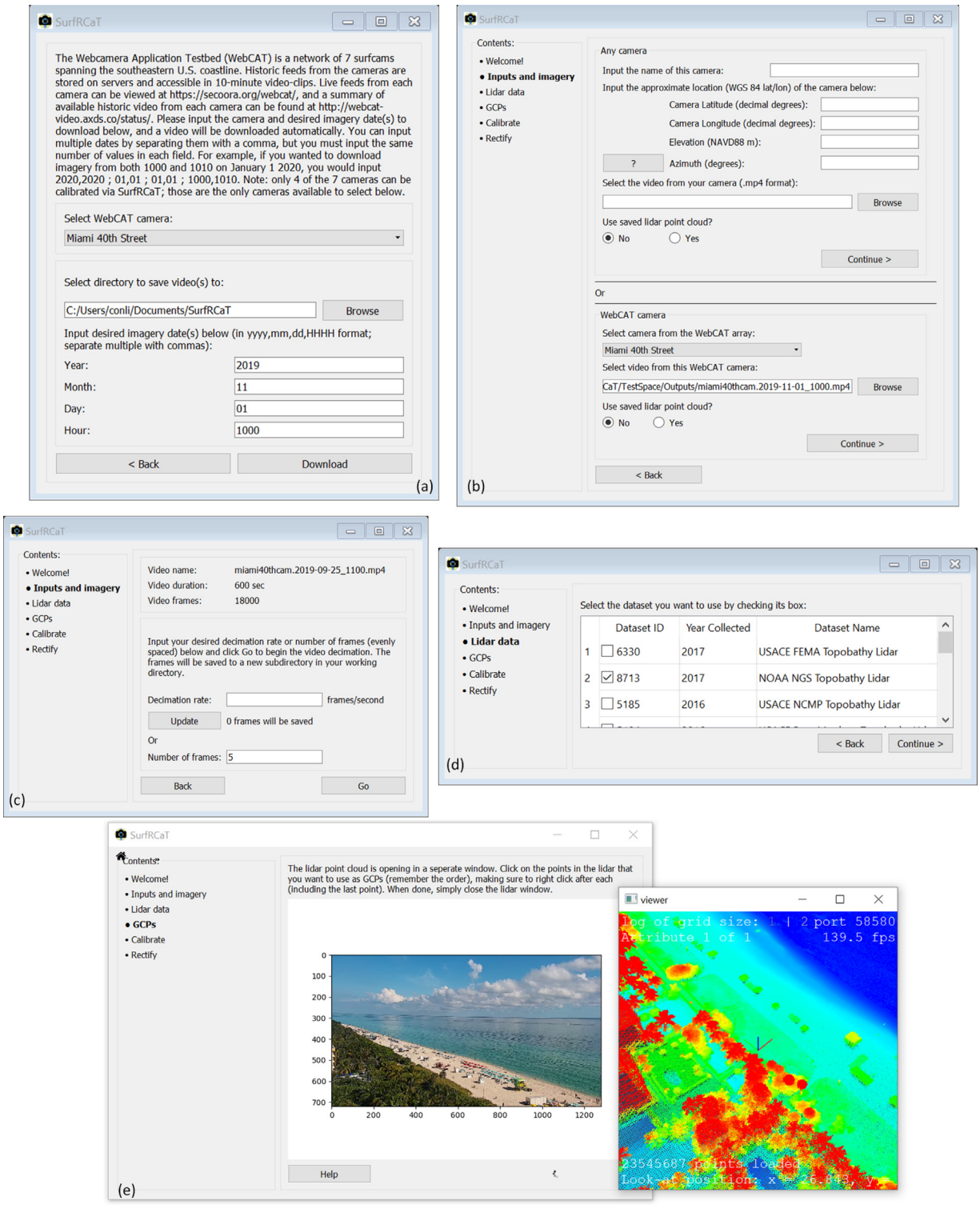

Fig. 2. Screenshots of the SurfRCaT user-experience for this illustrative example. (a) The WebCAT download utility. (b) The camera inputs window. (c) The frame extraction window. (d) Lidar dataset choice window. (d) The split screen view of the image and lidar data.

date(s)/time(s) for which they wish to download video(s). Videos as recent as ∼3 days prior to the current date can be downloaded.

3. Illustrative example

In this example, an image from one of the WebCAT cameras is rectified. Following the software documentation, the user downloads, installs, and invokes SurfRCaT. The user first downloads a 10-min video from November 01, 2019 at 1100 local time from the Miami 40th Street WebCAT camera by accessing and using the WebCAT Download Utility (Fig. 2a). The user then returns to the first window, chooses the camera calibration option, and establishes a working directory. Upon filling the fields in the "WebCAT camera" section of the next window (Fig. 2b), the user is prompted to input a frame extraction rate. In this case, the user

Fig. 3. (a) The calibration image from the WebCAT camera. (b) The rectified planimetric map of the image created using the SurfRCaT-derived calibration.

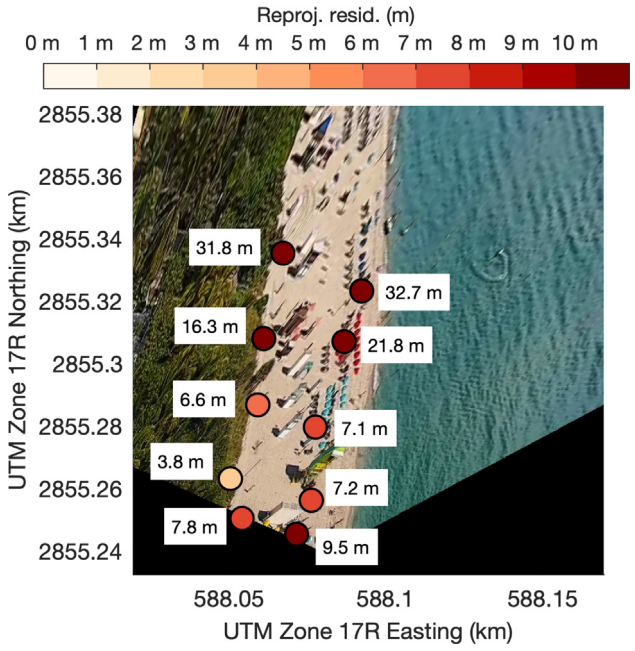

Fig. 4. The rectified planimetric map of the image shown in Fig. 3a (reproduced from Fig. 3b) with the reprojection residuals of 10 measured points superimposed.

chooses to extract five frames (Fig. 2c) from the video, and SurfRCaT extracts these frames and allows the user to scroll through them and choose one to use. The user then selects a pre-loaded lidar dataset to use in the GCP identification (Fig. 2d), a portion of which is subsequently saved locally. The user is presented with the lidar point cloud and the image and is able to see the same features in both after some manipulation of the point cloud (Fig. 2e). The user identifies (by clicking) at least three easily-distinguishable features in the lidar observations (e.g. lifeguard tower), closes the lidar viewer window, and identifies the same features (in the same order) in the image. This process is shown in Video1. Calibration is completed using these GCPs.

The saved calibration parameters are used to transform the extracted frames into rectified planimetric maps (Fig. 3). By downloading other videos from this camera with the WebCAT download utility, other images from this camera can then easily be extracted and rectified by selecting options to use the saved calibration parameters. Shoreline position and other quantitative parameters can be identified on these planimetric maps to track geophysical processes over a desired timescale.

3.1. Empirical evaluation

The accuracy of the rectified planimetric map in the illustrative example (Fig. 3b) is assessed using observations of 10 points within the field of view of the camera obtained via real-time kinematic (RTK) GPS. The reprojection residual of each point, that is, the distance between the measured location and location extracted from the rectified image, is computed (Fig. 4). The mean reprojection residual of the six points closest to the camera is 7.0 m, whereas the mean residual of the four farthest points is 25.6 m. In a geophysical context, both hurricane-induced and seasonal shoreline changes in Florida have been shown to often exceed 10 m in magnitude [28,29]. Thus, the southernmost 60 meters of the rectified image is likely of accuracy suitable to capture storm-induced and seasonal shoreline changes at this site. While this accuracy is relatively low compared to full on-site calibrations of research-grade cameras [6,9], this example illustrates that SurfRCaT can be used to obtain relevant and useful data from non-research cameras without a site visit. In general, the accuracy of SurfRCaT-derived rectified planimetric maps will depend on factors including remote GCP-availability and accuracy of user-input initial approximations. The documentation provides more guidance on this.

4. Impact

SurfRCaT could increase the number of potential quantitative coastal video-imaging stations in the U.S. by an order of magnitude. Such an increase has the potential to provide unprecedented validation for nationally-relevant coastal change models and tools such as the U.S. Geological Survey/NOAA Total Water Level and Coastal Change Forecast Viewer [30]. In addition to deepening our knowledge of existing questions by providing new study sites, SurfRCaT could also aid investigations into new research questions. For example, SurfRCaT is being applied to traffic cameras on the Outer Banks of North Carolina to examine the dimensions of storm-driven overwash deposits and dune breaches, as part of an ongoing investigation into the frequency of anthropogenic sand-moving events in the area [31]. Finally, the GUI front-end and application distribution of SurfRCaT, combined with its fully self-contained workflow, make it possible for groups beyond coastal researchers to perform camera calibrations and image rectifications as a first step toward monitoring local coastal changes.

Individual components of SurfRCaT also represent useful technological advances. The ability to automatically find and download (as a point cloud) airborne lidar datasets that cover a particular location could be useful for research applications beyond

remote-GCP extraction; for example, to perform a pre/post-storm topographic change study at a particular location. In this way, SurfRCaT could facilitate progress on existing questions such as storm-induced morphologic change processes. Further, the ability to display and query an airborne lidar point cloud without performing any programming could be a useful tool for research and non-research stakeholders alike.

5. Conclusions

The Surf-camera Remote Calibration Tool (SurfRCaT) enables potentially hundreds of already-operational surf-cameras on the United States coastline to be remotely calibrated, allowing the use of these cameras as quantitative coastal monitoring tools. SurfRCaT is distributed as a self-contained, GUI-driven application, making it usable by groups beyond academic coastal researchers who may lack programming knowledge or resources. This tool enables quantitative studies of coastal geophysical processes to be conducted at many new sites, facilitating study of both existing and new research questions, and provides functionalities useful for applications beyond its primary intended purpose.

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.

Acknowledgments

Funding was provided by the Southeast Coastal Ocean Observing Regional Association (SECOORA) through their annual student data challenge. The authors wish to thank Joe Long, Dave Foster, and Richard Conlin for valuable software testing and Justin Birchler and Kyle Kelso for conducting the validation camera calibration survey. The authors also wish to thank two anonymous reviewers whose comments greatly improved the quality of the software and manuscript. Any use of trade, firm, or product names is for discriptive purposes only and does not imply endorsement by the U.S. Government.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2020.100584.

References

[1] Wright LD, Short AD. Morphodynamic variability of surf zones and beaches: A synthesis. Mar Geol 1984;56(84):93–118. http://dx.doi.org/10.1016/0025-3227(84)90008-2.
[2] Vos K, Harley MD, Splinter KD, Simmons JA, Turner IL. Sub-annual to multi-decadal shoreline variability from publicly available satellite imagery. Coast Eng 2019;150:160–74. http://dx.doi.org/10.1016/j.coastaleng.2019.04.004.
[3] Pianca C, Holman R, Siegle E. Shoreline variability from days to decades: Results of long-term video imaging. J Geophys Res Oceans 2015;120:2159–78. http://dx.doi.org/10.1002/2014JC010329.
[4] Neumann B, Vafeidis AT, Zimmermann J, Nicholls RJ. Future coastal population growth and exposure to sea-level rise and coastal flooding - A global assessment. PLoS One 2015;10(3). http://dx.doi.org/10.1371/journal.pone.0131375.
[5] Holman RA, Stanley J. The history and technical capabilities of Argus. Coast Eng 2007;54:477–91. http://dx.doi.org/10.1016/j.coastaleng.2007.01.003.
[6] Holl KT, Holman RA, Lippmann TC, Stanley J, Plant N. Practical use of video imagery in nearshore oceanographic field studies. IEEE J Ocean Eng 1997;22(1):81–92. http://dx.doi.org/10.1109/48.557542.
[7] Holl KT, Holman RA. Wavenumber-frequency structure of infragravity swash motions. J Geophys Res Oceans 1999;104(C6):13479–88. http://dx.doi.org/10.1029/1999JC900075.
[8] 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.
[9] Holman RA, Symonds G, Thornton EB, Ranasinghe R. Rip spacing and persistence on an embayed beach. J Geophys Res Oceans 2006;111(C1). http://dx.doi.org/10.1029/2005JC002965.
[10] Nieto MA, Garau B, Balle S, Simarro G, Zarruk GA, Ortiz A, et al. An open source, low cost video-based coastal monitoring system. Earth Surf Process Landf 2010;35(14):1712–9. http://dx.doi.org/10.1002/esp.2025.
[11] 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.
[12] Andriolo U, Sánchez-García E, Taborda R. Operational use of surfcam online streaming images for coastal morphodynamic studies. Remote Sens 2019;11(1):1–21. http://dx.doi.org/10.3390/rs11010078.
[13] Conlin MP, A. Scheinkman. U.S. Surf-camera database, Zenodo, v1; 2020. http://dx.doi.org/10.5281/zenodo.3946697.
[14] Mole MA, Mortlock TR, Turner IL, Goodwin ID, Splinter KD, Short AD. Capitalizing on the surfcam phenomenon: a pilot study in regional-scale shoreline and inshore wave monitoring utilizing existing camera infrastructure. J Coast Res 2013;65:1433–8.
[15] Bracs MA, Turner IL, Splinter KD, Short AD, Lane C, Davidson MA, et al. Evaluation of opportunistic shoreline monitoring capability utilizing existing surfcam infrastructure. J Coast Res 2016;32(3):542–54.
[16] Valentini N, Balouin Y, Bouvier C. Exploiting the capabilities of surfcam for coastal morphodynamic analysis. J Coast Res 2020;95(sp1):1333–8.
[17] Wolf PR, Dewitt BA, Wilkinson BE. Elements of photogrammetry: with applications in GIS. 4th ed. New York: McGraw-Hill; 2014.
[18] Brignone M, Schiaffino CF, Isla FI, 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.
[19] Simarro G, Ribas F, Álvarez A, Guillén J, Chic Ò, Orfila A. ULISES: An open source code for extrinsic calibrations and planview generations in coastal video monitoring systems. J Coast Res 2017;335(5):1217–27. http://dx.doi.org/10.2112/JCOASTRES-D-16-00022.1.
[20] Sánchez-García E, Balaguer-Beser A, Pardo-Pascual JE. C-Pro: A coastal projector monitoring system using terrestrial photogrammetry with a geometric horizon constraint. ISPRS J Photogramm Remote Sens 2017;128:255–73. http://dx.doi.org/10.1016/j.isprsjprs.2017.03.023.
[21] Dusek G, Hernandez D, Willis M, Brown JA, Long JW, Porter DE, et al. WebCAT: Piloting the development of a web camera coastal observing network for diverse applications. Front Mar Sci 2019;6:353. http://dx.doi.org/10.3389/fmars.2019.00353.
[22] Riverbank computing limited. PyQt. 2019, https://wiki.python.org/moin/PyQt. [Accessed 01 April 2020].
[23] Hermann M. fman build system. 2019, https://build-system.fman.io. [Accessed 01 April 2020].
[24] NOAA office for coastal management. Lidar datasets at NOAA digital coast, FTP, 2020. http://ftp.coast.noaa.gov/pub/DigitalCoast/.
[25] PDAL Contributors. PDAL point data abstraction library. 2018, http://dx.doi.org/10.5281/zenodo.2556738, [Accessed 01 April 2020].
[26] Here technologies. pptk 0.1.1 documentation. 2018, https://heremaps.github.io/pptk/index.html. [Accessed 01 April 2020].
[27] Pyinstaller development team. Pyinstaller. 2019, https://www.pyinstaller.org/#. [Accessed 01 April 2020].
[28] Adams PN, Keough KM, Olabarrieta M. Beach morphodynamics influenced by an ebb-tidal delta on the north florida atlantic coast. Earth Surf Process Landf 2016;41(7):936–50. http://dx.doi.org/10.1002/esp.3877.
[29] Sallenger AH, Stockdon HF, Fauver L, Hansen M, Thompson D, Wright CW, et al. Hurricanes 2004: an overview of their characteristics and coastal change. Estuaries Coasts 2006;29:880–8. http://dx.doi.org/10.1007/BF02798647.
[30] United states geological survey, National oceanic and atmospheric administration, and national weather service. Total water level and coastal change forecast viewer. 2020, https://coastal.er.usgs.gov/hurricanes/research/twlviewer/. [Accessed 01 April 2020].
[31] Lazarus ED, Goldstein EB. Is there a bulldozer in your model? J Geophys Res Earth Surf 2019;124:696–9. http://dx.doi.org/10.1029/2018JF004957.