CoastalImageLib: User Manual

Maile McCann, et al

Coastal and Hydraulics Laboratory
U.S. Army Engineer Research and Development Center
Duck, NC

Abstract

This user manual is intended to provide documentation and instruction on utilizing the CoastalImageLib python library for processing coastal imagery and creating common coastal image products.

1. Introduction

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, create subsampled pixel instruments for use in bathymetric inversion, surface current, run-up calculations, as well as 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 software such as the Argus tower in Duck, NC and mini- Argus stations throughout the U.S. 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.

2. Software Requirements

2.1. Programming Language

- Python version 3.6 or later

2.2. Required Packages

- numpy
- OpenCV
- scipy
- imageio
- yaml
- sys
- skimage

2.3. Required Packages (support functions only)

- yaml
- datetime
- argusIO (provided in CoastalImageLib)

3. Data Requirements

3.1. Known Inputs

- Camera intrinsic values
- Camera extrinsic values
- Local origin and angle (if coordinate transform is required)

4. Workflow

4.1. corefunctions.py

This module contains the core functions for merging multiple cameras and georectifying oblique images onto a user- defined real world XYZ grid. This workflow was in part adapted from capabilities in the CIRN Quantitative Coastal Imaging Library [3], the ARGUS Coastal Imaging System [1], and the USGS CoastCam System.

The workflow of rectification functions is as follows:

1. Define a grid using **XYZGrid**(xlims, ylims, dx=1, dy=1, z=0). Specify x and y limits and resolution of the grid in x and y directions. The value given for z should be equal to the esimated water level at the time of data collection relative to the local vertical datum used in specifying extrinsic information.
2. Initialize the recifier object using **Rectifer(XYZGrid)**. The argument **XYZGrid** is your rectification grid from step 1. The resultant object is valid for any rectification using the same xyz grid. However, for subsequent rectification tasks, check that the z value is still correct.
3. Call **mergeRectify**(images, intrinsic_list, extrinsic_list, coords = 'local', origin = 'None', mType = 'CalTech')
   - This function supports intrinsics formatted in CalTech convention or in DLT convention, and extrinsics in geo coordinates or local
   - **mergeRectify()** is designed to merge and rectify multiple cameras at one timestamp into a single frame
   - For multiple timestamps, loop through **mergeRectify** and rectify each desired frame
   - You do not have to re- initialize the **XYZGrid** object or the **Rectifier** object for each frame on the same grid, merely call **mergeRectify()** with your new set of images

pixelproducts.py contains functions to generate pixel instruments from a given set of images for use in bathymetric inversion, surface current, or run-up calculations.

imageproducts.py contains functions to generate statistical image products for a given set of images and corresponding camera extrinsic and intrinsic vales. These image products are timex, brightest, variance, and darkest.

4.2. supportfunctions.py

supportfuncs.py contains supporting functions to format intrinsic files, convert extrinsic coordinates to and from geographical and local coordinate systems, calculate extrinsic values, and DeBayer .raw Argus files into .avi files collected from the Argus tower [1], utilizing functions contained in **argusIO.py**. The module **argusIO.py** includes functions for further utilizing .raw Argus files, however will not be further discussed in this paper as they don't apply to data collected outside of the Argus system. They are included in the library for ease of use for Argus specific applications.

5. Intrinsics

The intrinsic vector is defined, in both CIRN convention and in CalTech camera calibration results, as the following:

CIRN Intrinsics Variable	Caltech Variable	Description
intrinsic_list[0]	nx	Number of pixel columns
intrinsic_list[1]	ny	Number of pixel rows
intrinsic_list[2]	cc(1)	U component of principal point, defined from top left corner of image
intrinsic_list[3]	cc(2)	V component of principal point, defined from top left corner of image
intrinsic_list[4]	fc(1)	U components of focal lengths (in pixels)
intrinsic_list[5]	fc(2)	V components of focal lengths (in pixels)
intrinsic_list[6]	kc(1)	Radial Distortion Coefficient
intrinsic_list[7]	kc(2)	Radial Distortion Coefficient
intrinsic_list[8]	kc(5)	Radial Distortion Coefficient
intrinsic_list[9]	kc(3)	Tangential Distortion Coefficient
intrinsic_list[10]	kc(4)	Tangential Distortion Coefficient

5.1. CalTech Camera Calibration

For applications where the camera intrinsics are unknown, the user is directed towards the CalTech Camera Calibration library: http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/ref.html.

Note: Matlab is required.

From the library homepage: *This is a release of a Camera Calibration library for Matlab with a complete documentation. This document may also be used as a tutorial on camera calibration since it includes general information about calibration, references and related links.*

The list of internal parameters:

- Focal length: The focal length in pixels is stored in the 2x1 vector fc.
- Principal point: The principal point coordinates are stored in the 2x1 vector cc.
- Skew coefficient: The skew coefficient defining the angle between the x and y pixel axes is stored in the scalar alpha_c.
- Distortions: The image distortion coefficients (radial and tangential distortions) are stored in the 5x1 vector kc.

5.2. Direct Linear Transform / Walton m- vector

Direct linear transformation (DLT) is an algorithm which solves a set of variables from a set of similarity relations [5]. In camera calibration, this yields a camera matrix in Walton m- vector notation that implicitly includes both intrinsic and extrinsic values. When utilizing DLT for camera calibration, specify in the mType tag during $mergeRectify()$. Extrinsic values can be passed through as an empty vector.

Eg: **mergeRectify(self, images, intrinsic_list = m, extrinsic_list = [], coords = 'local', origin = 'None', mType = 'DLT')**, where m = the DLT coefficient vector A-L.

X	X position of camera
Y	Y position of camera
Z	Z position of camera
azimuth	The horizontal direction the camera is pointing and positive CW from z axis.
tilt	The up/down tilt of the camera. 0 is the camera looking nadir, +90 is the camera looking at the horizon right side up.
roll	The side to side tilt of the camera. 0 degrees is a horizontal flat camera. Looking from behind the camera, CCW would provide a positive swing.

6. Extrinsics and Coordinate Systems

The extrinsic vector is defined as:

$$extrinsic_list = [X, Y, Z, azimuth, tilt, roll] \tag{1}$$

where

Coordinates can be input in either local (user defined) or geographical. However, in order to work, the camera extrinsics and points in question need to be in the same coordinate system. You cannot use the projective matrix on geographic points when extrinsic are defined in local and vice versa [1].

The following diagram depicts the CIRN convention for transforming geographical to local coordinates. The subscripts WG stand for world geographic, and the subscripts WL stand for world local.

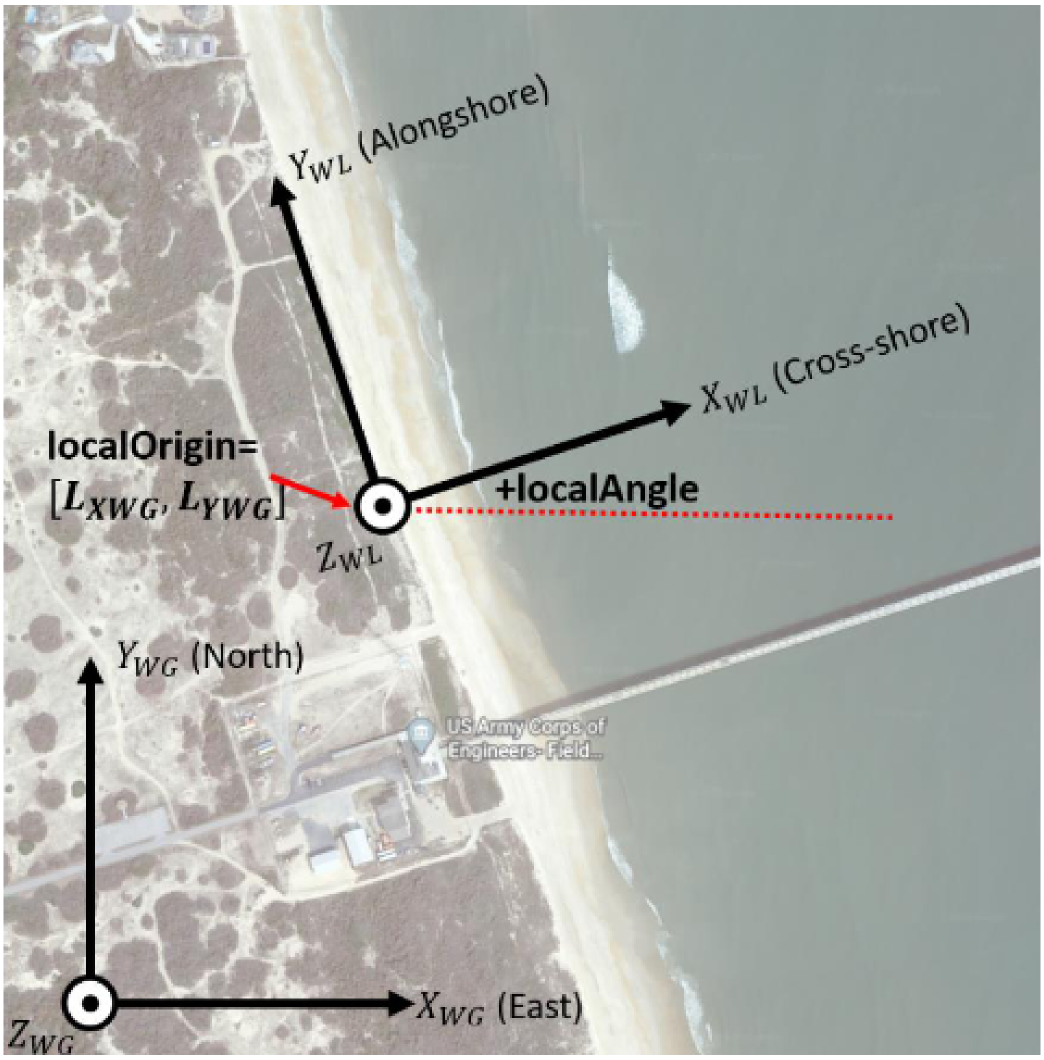

Figure 1: Relationship between World Coordinate Systems, Geographical and Local, from CIRN Quantitative Coastal Imaging library [1]

References

[1] Holman R.A., Stanley J. "The history and technical capabilities of Argus". Coastal Eng, 54 (6) (2007), pp. 477-491, 10.1016/j.coastaleng.2007.01.003

[2] Holman, R.; Haller, M. "Remote sensing of the nearshore." Annu. Rev. Mar. Sci. 2013, 5, 95–113.

[3] B. L. Bruder and K. L. Brodie, "CIRN Quantitative Coastal

Imaging library," SoftwareX, vol. 12, p. 100582, 2020, doi: https://doi.org/10.1016/j.softx.2020.100582.

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

[5] Abdel-Aziz, Y.I.; Karara, H.M. "Direct Linear Transformation from Comparator Coordinates into Object Space Coordinates in Close-Range Photogrammetry". Photogrammetric Engineering Remote Sensing. 81 (2): 103–107. doi:10.14358/pers.81.2.103

[6] Bradski, G. The OpenCV Library. Dobb's J. Softw. Tools 2000, 3, 1–81.