How to run Version 2.0 and some benchmarking of the results.

The following parameters need to be added to the input file argus02b.m:

params.DECIMATE = 1;

params.kappa0 = 2; % still used in bathyFromKAlpha (phase 2)

params.nlinfit = 1; % versus LMFnLSQ

The call to run cBathy is the following:

bathy = analyzeBathyCollect(xyz, t, data, cam, bathy);

The inputs are:

xyz : should be number of points x 3 – ( N\_pts x 3 )

t : time vector, in seconds, should be 1 x number of times – ( 1 x N\_t )

data : image stack should be number of times x number of points – ( N\_t x N\_pts )

cam : array of indices associating pixels with cameras (all 1’s for single camera) – (N\_pts x 1)

bathy : input parameters structure created by argus02b

* bathy.params is a structure containing the following fields (all set in argus02b):

stationStr – ‘argus02b’

dxm – output grid spacing ‘X’ (12.5)

dym – output grid spacing ‘Y’ (25)

xyMinMax – grid range ([min\_x , max\_x , min\_y , max\_y] ) [80 800 0 1500]

tideFunction – ‘cBathyTide’

MINDEPTH – 0.25

QTOL – 0.5

minLam – 10

Lx – tile size ‘X’ – 37.5

Ly – tile size ‘Y’ – 75

maxNPix – maximum number of pixels in a single tile – 80

minValsForBathyEst – minimum number of pixels required in a single tile – 4

DECIMATE – 1

kappa0 – smoothing factor – 2

nlinfit – set to 1 (indicating to use nlinfit as the LSQ option)

fB – frequency bins – row vector

nKeep – 4

debug – structure

offshoreRadCCWFromx – 0

Running 4 data from cameras (visible and lwir, wide and narrow fields of view), this routine takes about 4 minutes (~60 seconds per file).

Sample results from 26 June, 2019:

|  |  |  |  |
| --- | --- | --- | --- |
| Camera | Median Error (m) | Mean Error (m) | Sigma (m) |
| Visible WFOV | 0.46 | 0.56 | 0.6 |
| Visible NFOV | 0.24 | 0.27 | 0.75 |
| LWIR WFOV | 1.0 | 2.0 | 3.2 |
| LWIR NFOV | 0.7 | 1.3 | 1.9 |