**Overview of RHIB\_ADCP\_processing package**

|  |  |  |  |
| --- | --- | --- | --- |
| **Top level function/script** | **1st sub-level function** | **2nd sub-level function** | **Description** |
| RHIBproc |  |  | File manager:   * Looks for raw files * Processes data and saves files * Makes and saves a few figures |
|  | ParseDeployment |  | Parsing manager |
|  |  | parse\_imu | Parses IMU   * IMU data is not currently used or stored |
|  |  | parse\_rdi\_adcp | Parses ADCP data from RDI instruments |
|  |  | parse\_gps | Parses data from Hemisphere GPS |
|  | ProcADCP |  | Processing manager   * Aligns GPS and ADCP times * Transforms to Earth coordinates * Removes boat velocity |
|  |  | nucRepair | Patches minor error in retrieving RHIB computer timestamps |
|  |  | compute\_vessel\_vel | Gets boat location and velocity from GPS |
|  |  | gps\_ltln2vel | Computes vessel velocity if vessel\_vel\_method=’GPRMC distance/time’ |
|  |  | gps\_line\_interp | Matches GPS heading entries with GPS time |
|  |  | adcp\_beam2earth | Rotates velocity to Earth coordinates |
|  | ClipAirTime |  | Removes leading/trailing portions of record collected in air |
|  | PlotTrack |  | Plots boat track |
|  |  | Draw\_LeConteCoastline | Adds simple coastline to map |
|  | PlotBeams |  | Plots timeseries of beam data |
|  | PlotENU |  | Plots timeseries of processed velocities |
| Interactive\_LocateBottom |  |  | Produces an interactive figure for identifying bottom contours   * User can adjust identification thresholds and visualize results * Allows for manual exclusion of backscatter data misidentified as a hard surface |
|  | LocateBottom |  | Uses peak in backscatter to identify likely bottom contours |
|  |  | Echo2Backscatter | Estimates backscatter from echo intensity   * Accounts for beam spread and water absorption |
| CreateQCMasks |  |  | Creates masks based on thresholds for correlation, turn rate, bottom contours, etc. |
| LocateFeature |  |  | Displays interactive figure for user to explore location of features seen in beam data |
|  | LoadDeployment |  | Loads processed file that user selects from list |
|  | AddTerm |  | Displays a list of terminus files for user to select from   * Terminus line will be displayed on map |
|  | adcp\_earth2beam |  | Translates boat velocity into beam coordinates   * Used to remove boat motion from beam data |
|  | beamlocations |  | Calculates location of each observation based on beam angle, depth, tilt, and heading |
|  | Draw\_LeConteCoastline |  | Adds simple coastline to map |
| DefineTransects |  |  | Interactive figure for dividing deployment into transects   * Automatically saves file with transect endpoints |

**How to process ADCP data from RHIBs**

* Open RHIBproc.m script in Matlab

A screenshot of a computer program

Description automatically generated

The script heading gives a brief summary the basic steps to run the script successfully, which are detailed below.

* Change directories to the folder that contains the raw data folder **OR** change the basepath as described in the next step

A white rectangular object with a black border

Description automatically generated

* Customize processing details as desired

A screenshot of a computer program

Description automatically generated

Most common customizations:

basepath='/Volumes/sci2023/data/'; Access files not in working directory

overwrite=true; Use this when reprocessing existing files

minadcp=X; mingps=Y; Set minimum file sizes for processing (to exclude deployments where the boat didn’t actually go in the water or to only pull GPS data if the ADCP was off).

addterm=0; Leave old terminus line off map if it is not helpful

test=1; Test out different adjustments to heading offset

serial=[xxx]; Add a serial number to the list if using a new instrument

offset=[xxx]; Set heading offset for each instrument used

\*note: If instrument offset changes between deployments, it might be better to run the parsing and processing functions outside of this script (see README)

* Run the script

A computer screen with a blue border

Description automatically generated

A prompt will appear in the command window to verify the offsets for each instrument.

* #14158 is the Workhorse and #24653 is the Sentinel V
* n will abort the script so that you can fill in the correct values as shown in previous step
* y will continue with the script, processing all files that do not have a corresponding processed file (or reprocessing existing files if overwrite=true)

During processing, progress updates will appear in the command windowA screenshot of a computer

Description automatically generated

The figures being created will be displayed briefly before they are saved

An additional message will let you know when all files have been processed

A screenshot of a computer

Description automatically generated

Processed files will be saved to a separate folder with a similar file structure

A screenshot of a computer

Description automatically generated

**What else can this package do?**

**Convenient file loading**

* Run LoadDeployment()

adcp=LoadDeployment(basepath)

A screenshot of a computer

Description automatically generated

basepath should be the directory that contains the processed data folder (the same directory used for processing)

**Define transects**

* Open DefineTransects.m script in Matlab

A screenshot of a computer

Description automatically generated

basepath should be the directory that contains the processed data folder (the same directory used for processing)

* Run the script
  + The script runs LoadDeployment, so select file to load as above
* An interactive figure will display

A screenshot of a computer

Description automatically generated

* The panel on the left will display the tracks for this deployment colored by time
  + The colorbar displays the corresponding hour:minutes
  + Use the axes tools to zoom/pan to the desired portion of the deployment
* Initially, there will be two boxes on the right
  + These boxes will auto-populate with start and end times for each transect
* Define start and end points for each transect by clicking on the boat tracks
  + Zoom in enough that the point you wish to select is distinct from nearby points
    - The algorithm will select the data point that is nearest in space to the location that you click on the plot
    - Deselect axes tools to enable point selection (i.e., if zoom tool is selected, clicking on the axes will only zoom)

A screenshot of a computer

Description automatically generated

* + Points must be selected in time order (start at the blue end)
  + The first point selected will designate the start of a transect
    - A green dot will appear on top of the point selected and the start box will populate with the datenum of that point (this is just to show that a point has been identified)

A screenshot of a computer

Description automatically generated

* + The next point selected will designate the end of a transect
    - A red dot will appear on top of the point selected, the finish box will populate with the datenum of the point, and two new boxes will appear

A screenshot of a computer screen

Description automatically generated

* + Continue selecting points until all transects have been defined

A screenshot of a computer

Description automatically generated

* + If satisfied with the point selections, click ‘Save and Exit’
    - Times of endpoints and indices included in each transect will be saved to a new file in the same folder as the ADCP data
    - Figure will automatically close
  + If the figure is closed without clicking ‘Save and Exit’, all endpoints will be lost

**Quick and dirty identification of bottom contours**

* Run LocateBottom() with default thresholds

adcp=LocateBottom(adcp);

* This adds the bottom\_mask field (among others) to the adcp structure
* contains a mask for each beam, multiply them together to get a single mask to apply to each component of the processed velocity

botmask=adcp.bottom\_mask(:,1,:).\*adcp.bottom\_mask(:,2,:).\*adcp.bottom\_mask(:,3,:).\*adcp.bottom\_mask(:,4,:).\*adcp.bottom\_mask(:,5,:);

* It is expected that this function with the default thresholds will miss some bottom contours and identify some random noise as bottom contours
  + **Use this for plotting only; do not save results as processed data**
* Thresholds can be adjusted via argument inputs
* For a more accurate identification of bottom contours, Interactive\_LocateBottom() should be used (this is generally too involved to use in the field)
* PlotENU() can be used to quickly plot the revised velocity.
  + To do this, you’ll want to apply the bottom mask to the velocity field
  + First, repeat the bottom mask so that it is the same shape as the velocity field

botmask=[botmask botmask botmask botmask];

* + Then apply it to the velocity field within the adcp structure

adcp.vel=adcp.vel.\*botmask;

* + Finally, feed the adcp structure into the function

[fig,ax]=PlotENU(adcp)

* + Note that a second input to PlotENU can be used to designate a range of indices to plot

e.g. the transect indices identified by DefineTransects could be used to plot a specific transect (see LoadDeployment for how to load the transect info along with adcp).

Before:

A screenshot of a computer generated image

Description automatically generated

After:

A screenshot of a graph

Description automatically generated

**Apply some basic quality control**

* Run CreateQCMasks() specifying a turn rate threshold
* By default, this masks out bottom 10% of water column and data with correlation below 64
* This function requires that bottom contours have already been identified

QC=CreateQCMasks(adcp,turn=.15);

* QC contains combo\_mask (the combination of masks for each threshold), which has masks for each beam
  + Multiply them together to get a single mask to apply to each component of the processed velocity

QCmask=QC.combo\_mask(:,1,:).\*QC.combo\_mask(:,2,:).\*QC.combo\_mask(:,3,:).\*QC.combo\_mask(:,4,:).\*QC.combo\_mask(:,5,:);

* Errors in bottom identification will carry through to the quality control mask
  + **Use this for plotting only; do not save results as processed data**
* PlotENU() can be used to quickly plot the revised velocity
  + Refer to the bottom identification section above for how to apply a mask to the velocity field and use it in PlotENU

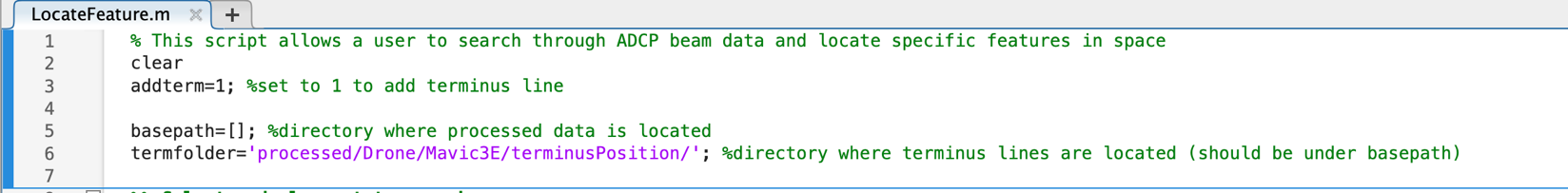
After:

A screenshot of a graph

Description automatically generated

**Identify features in beam data to locate in space**

* Open LocateFeature.m in Matlab



* Identify folders in which to find data
  + Set addterm=0 if no terminus lines have been identified
* Run the script

A screen shot of a computer

Description automatically generated

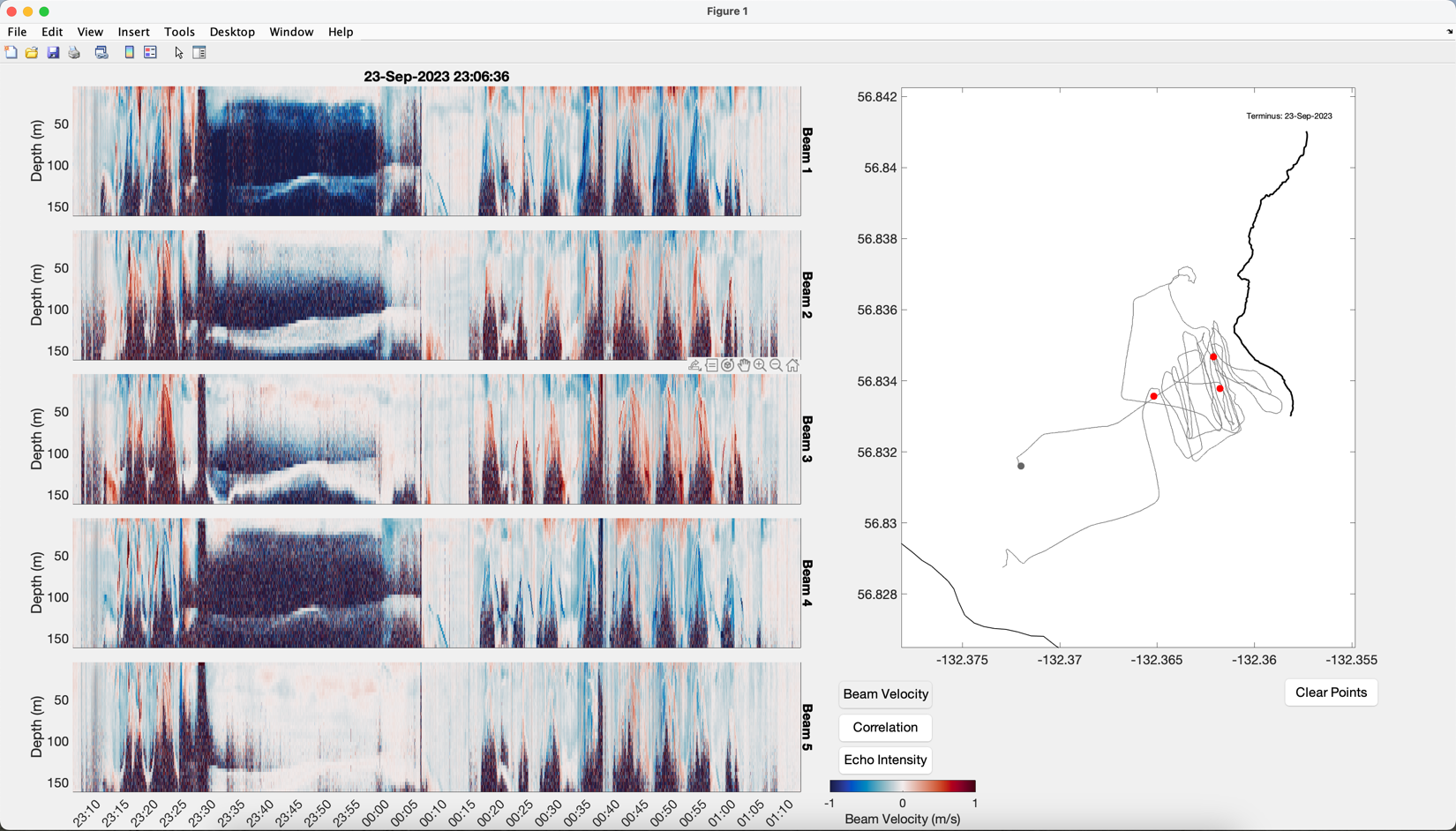
You will be prompted to select a file to load

A screenshot of a computer

Description automatically generated

If you set addterm=1, it will display the time that the selected deployment begins and prompt you to select a terminus line to display on the map

The interactive figure will then be displayed:



Click here to select which timeseries data to display

Click anywhere in the timeseries data and the location of that data point will be shown on the map

Use tools to zoom or pan within the timeseries data (Deselect tool to select a point to display on map)

Location of selected data point(s)

Click here to clear selected data points

This figure is solely for exploration. Close figure when finished.