State of the Ecoson's refactorization

NumSeaHy

November 20, 2024

Todo list

The charge of the echosounder data is done in the original Octave version of the Ecosons using the scripts present `https://github.com/daniel-rperez/ecosons/tree/master/ecosons_lib/formats`. Inside the formats folder there are the following scripts:

- `fmt_lowranceSL1.m`
- `fmt_lowranceSL2.m`
- `fmt_lowranceSLG.m`
- `fmt_simradDG.m`
- `fmt_simradRAW.m`
- `parseXML.m`

Here, the biggest module is the `fmt_simradRAW.m` that contains load function for data saved in RAW3 format. Actually this function has been has been refactorized in Julia with the name `EA640.jl` that accepts uniquely data from the EA640 echosounder. A desirable feature is extend the module to accept not only EA640 data files, but also EK80 datafiles. The loading code of the data corresponding to the lowrance is much shorter, but not has been refactorized. Another interesting feature to implement is try to generate a function that saves all the data loaded the first time in a `.JLD2` file, instead have to load the binary files everytime that one have to work with the data. In this way, one only has to load the binary file (relatively high computational cost if a lot of .raw files are present) once, then they are saved in .JLD2 files and the following times the date are loaded from .JLD2 files, which is much faster .

Add support for load datafiles correspded to the EK80 echosounder

Add JLD2 loading functionality

Once the data is loaded, the next step is compute the bathymetry associated with these data loaded. First, the seabed bottom based on these data should be dettected. For this two different methods saved in the following folder of the Octave code `https://github.com/daniel-rperez/ecosons/tree/master/ecosons_lib/procs` are available:

- `getFirstHit.m`
- `getAverageHit.m`

This two functions has been refactorized in Julia and are inside the `ComputeBathymetry.jl` module with the names `getFirstHit` and `getAverageHit` respectively. Although that several test has beem performed to check if this two refactorized algortihms gave the same results as the Octave ones, would be desirable to perform more tests an see if there is something to correct in the Julia code .

Perform more tests in the bottom detection algorithms

Once the bottom has been detected with one of the two previous methods, a smoothing proccedure should be applied to the data. The function which performs this operation is also located in the same folder that the previous two methods https://github.com/daniel-rperez/ecosons/tree/master/ecosons_lib/procs and has the following name:

- `smoothRange.m`

This function has been also refactorized into Julia with in the same module where the functions to locate the bottom line are located: `ComputeBathymetry.jl` and has the name `smoothRange!` since in Julia is common to put a ! sign if the function modifies the data passed to it.

The next step consists in apply a tide correction to the computed data based on the data of the state of the tide. The tide correction in the original octave code, is performed using the following file https://github.com/daniel-rperez/ecosons/blob/master/ecosons_lib/procs/tidecorrect.m and the menu that display the different option to the user in the following location: https://github.com/daniel-rperez/ecosons/blob/master/ecosons_bathy/ec_bathymetry_tidecorrection.m that gives to the user several option to compute the tide correction:

- [1]: Input manually the tide times and heights.
- [2]: Input day tide height filename.
- [3]: Automatically choose a tide height file.

In the Julia refactorization the option 2 is the only available one and is inside the `Tides.jl` module with the following name: `tidecorrectDay!`, the tide data are given in a `.dat` file with the information corresponding to a day like follows:

hour	minute	height[m]
5	46	3.1
11	42	0.8
15	57	3.53
23	12	0.76

the funtion corresponding to the option [1] has been refactorized with the name `tidecorrect!`, but the enter of the input user data function is not refactorized yet, so would be desirible to extend the functionality and let to the user either introuduce manually the data with an interactive menu or choose one of all the tide file between all the present tide files .

Let to the user introduce the data or choose a concrete tide file

The next step consists in perform a bathymetry subsampling, this method allows to introduce the effects of the movement of the ship into the obtained results by applying a filter of minimums. This is performed using the content contained in the following Octave file https://github.com/daniel-rperez/ecosons/blob/master/ecosons_bathy/ec_bathymetry_subsampling.m and the function that performs the Subsamplig is called radialSubsampling that is located in the following octave file: https://github.com/daniel-rperez/ecosons/blob/master/ecosons_lib/procs/radialSubsampling.m. The Julia refactorization of this functionality is inside the module `Subsamplin.jl` and the function that perform the operations is called `ec_bathymetry_subsampling!`. Only a very little quantity of examples to see if the refactorization was correct were performed (this functionality wasn't a priority), so would be desirable to perform more test to correct the refactorization if neccesary.

Check the refactorization of the Subsampling module

The next step consist in perform a resampling of the data by using a Monte Carlo simulation, arounf each of the bathymetric points generating ten points gaussianaly distributed. This operation is performed by using the content contained in https://github.com/daniel-rperez/ecosons/blob/master/ecosons_bathy/ec_bathymetry_resampling.m

The program has three options to perform this computation:

- [1] External GIS application.
- [2] Kernel derivative method.
- [3] No slope correction.

Once that the slopes are computed with one of the three methods the resampleBathymetry function that is located in this folder: https://github.com/daniel-rperez/ecosons/blob/master/ecosons_lib/procs/resampleBathymetry.m is called.

The option [1] is not available in Julia, since I didn't have enough time to learn how to import and export the data to the GIS software. The only refactorized option to Julia was the option [2] and it was not intensely tested. The refactorization was splited in two main modules: `Resampling.j` and `Slopes.jl`. The second one compute the slopes, and the first one uses they to perform the resampling method. This modules has not been tested so would be desirable to check the functioning of them . At the same time, would be also useful to implement the method [3] when no slope correction is applied .

Verify and correct the Kernel derivative method

Implement a resampling with no slope correction

The last step is display the results obtained in a way that can let to the user extract some valuable information. For this pupose, the original Octave formulation has the following funcionalities:

- [1] Plot Echogram: that plots the echogram of a given transect. The code that generates this plot can be found in the Octave code here: https://github.com/daniel-rperez/ecosons/blob/master/ecosons_bathy/ec_plot_echobottom.m
- [2] Plot Transects: that plot the paths covered by the ship in each of the transects. The code that generates this plot can be found in the Octave code here: https://github.com/daniel-rperez/ecosons/blob/master/ecosons_bathy/ec_plot_transects.m
- [3] Plot 3-D bathymetry: that plots the transects in 3D where the z-component corresponds with the bathymetry height. The code that generates this plot can be found in the Octave code here: https://github.com/daniel-rperez/ecosons/blob/master/ecosons_bathy/ec_plot_bathymetry.m
- [4] Plot Bathymetry transect crosses: that allows to represent the bathymetric differences between transect crossings. The image is only displayed if the included transects in the computation of the bathymetry present crossings. The code that generates this plot can be found in the Octave code here: https://github.com/daniel-rperez/ecosons/blob/master/ecosons_bathy/ec_plot_bathycross.m
- [5] Interpolated bathymetry: this funcionality allows to generate much more quality images by applying an interpolation method based on a ponderate distance method. The interpolation method in the Octave code can be found here: https://github.com/daniel-rperez/ecosons/blob/master/ecosons_lib/utils/trinterpmap.m. All the availables plots are the three following ones, and can be found here: https://github.com/daniel-rperez/ecosons/blob/master/ecosons_bathy/ec_plot_interpolation.m:
  - [1] Colored map representation: that represents in a 2D image the processed bathymetry data.
  - [2] 3-D elevation: that represents in a 3D image the processed bathymetry data .
  - [3] Contour map: that represents all the bathymetric lines together extracted from the height of each of the transects.

To replicate the interpolation funcionalities a function that performs the same interpolation that the Octave code has been created inside the `Utils.jl` file with the name `trinterpmap`. Even that this funciton seem to return correct results will be desirable make some other test to verify the correct behaviour of the function .

Desirable to make the previous three items using Makie

To replicate the Octave ploting funcionalities, a module in Julia called `Plots` inside the `Plots.jl` file has been created. Some of the Octave function this module has the following function that replicates the behaviour of the Octave funcionalities:

Verify the correct performance of the `trinterpmap` function

- `ec_plot_echobottom`
- `plot_ping`
- `plot_bathymetry_line`
- `plot_transect`

The 3D plots of the bathymetry has been created by using Plotly, that allows to generate interactive plots in an easy way. This code has not been generated here, has been generated in the BathymetryApp https://github.com/NumSeaHy/SeabedExploration/blob/main/src/BathymetryApp/App.jl and would be desirable to fill the Plot module with this funcionality .

Fill the Plot module with 3D plots generated by Plotly or Makie