Compute pressure fields and forces from PIV

Author: Nils Tack
Compiled with Matlab R2020a

Script used to prepare PIV data files, create masks and outlines from images, manually correct outlines (if necessary), compute experimentally derived pressure fields, thrust and drag forces acting on the body, and the Froude efficiency.

Please refer to the following publication for details about pressure field calculations from 2D velocity fields: Dabiri, J. O., Bose, S., Gemmell, B. J., Colin, S. P. and Costello, J. H. (2014). An algorithm to estimate unsteady and quasi-steady pressure fields from velocity field measurements. J. Exp. Biol. 217, 331–336.

I. Things to consider before using this script

This Matlab Script was designed to compute forces for animals swimming 'upward' in the field of view. Though the swimming direction has no influence on pressure fields computations, it does have an importance when determining the nature of the forces at the animal/fluid interface. This script was specifically written with the assumption that thrust points upward (toward positive values) while drag points downward (toward negative values). This orientation is the most logical since it does not require any sign change for computational and plotting purposes. It is also more intuitive when thinking about animals moving forward.

To compute pressure fields and forces accurately and reliably, it is essential to obtain good quality velocity fields. Though this consideration appears obvious, many issues can arise from improper velocity fields, and vector fields that are not matching the values expected along and outside of the domain boundaries. This tutorial describes situations that can arises when computing velocity fields using DaVis. Other outcomes may exist if using another software (e.g., PIVlab). Three cases must be considered when validating the suitability of the velocity fields to compute pressures:

1. A still tank of water

The domain boundaries are fixed and are defined by the edges of the tank. Therefore, the velocity field on the edge, and outside of the domain should all be equal 0. This is critical for proper pressure calculation since any change in velocity from free-stream velocity to 0 will indicate that water is stopped by a solid surface. Otherwise, pressure will be calculated outside of the domain and lead to major inaccuracies. If the edges of the tank are <u>all</u> visible, make sure to mask them during PIV computation. This is generally performed by assigning a 0 value for the x and y components of each vector falling outside of the domain. This Matlab script will not read-in NaN as 0.

2. A swim tunnel or partial view of a still tank

In this situation, it must be assumed that water is flowing in and out of the field of view, without being constrained by a completely closed domain. In this situation, it is critical to **never** mask the inlet/outlet since this would generate an "artificial" wall by causing a sudden change in velocity along the edge(s) of the mask that would otherwise not exist. One may assume that the edge of the frame will still produce this result, but this is not the case. As long as the processing parameters used to compute velocity fields are chosen appropriately, the velocity of the water entering and leaving the domain can be accurately calculated along the edges of the frame. It is therefore crucial to verify that velocity vectors are assigned and receive non-0 values for the interrogation windows located along these 'open' sections of the domain. However, like the previous case, any solid boundaries must be masked. In the following example (fig. 1), when considering a flume, water is flowing from the left side of the frame to the right. However, we can still see the top and bottom walls of the flume in the field of view. In this case, the left and right sides of the frame remained unmasked, while the top and bottom sections are masked to define the solid boundaries of the flume (also defining the solid edges of the domain). The same principle can be applied to a still tank of water for which not all walls are in the field of view, regardless of their number, orientation, and position.

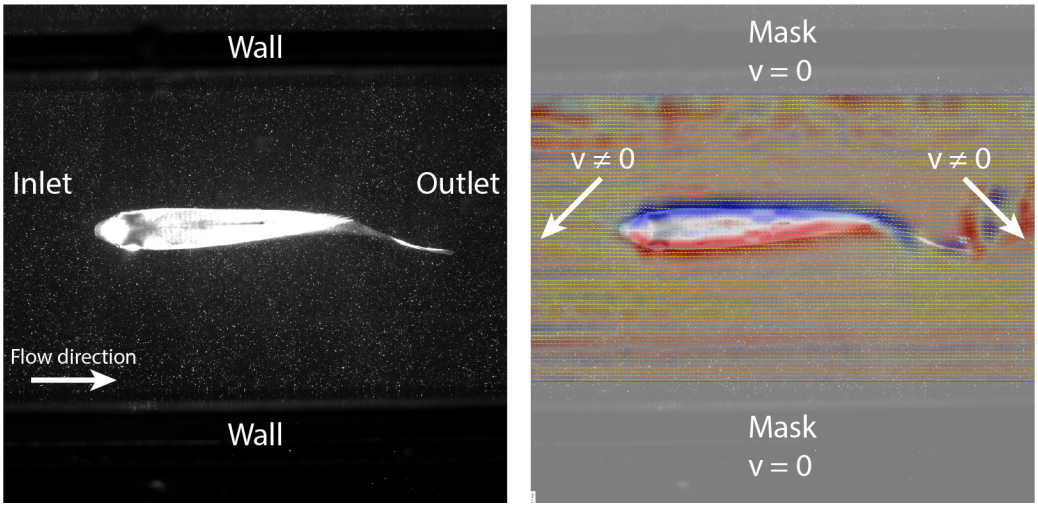

Figure 1. Example of PIV performed in a flume to illustrate how masking should be performed in DaVis to avoid inaccurate pressure field calculations.

3. No boundaries

This scenario generally applies to large tanks or close-up recordings. Similar to the previous case, it must be assumed that water is entering and leaving the frame. However, here, water is free to flow through the four 'open' edges of the frame, thus prohibiting any alterations of the domain boundaries. In this case, no masking is possible. The only practical way to "mask" the frame to, for example, reduce processing time due to the computation of large empty spaces is to directly crop the original video frames. However, a proper experimental setup should preclude this situation all together. Another method consists in attributing NaN to interrogation windows falling outside of the desired area to be subsequently discarded before performing pressure calculations.

Keep in mind that this procedure applies to computations performed using DaVis and may not hold true with other PIV software.

II. Getting started with the script

Before running the main function called **'MasterComputePressureWithMask'** make sure that the following folders are empty and exist in the **Data** folder: forces, images, masksBW, outlines, PIVclean, PIVoriginal, and pressure.

1. Loading PIV data and image sequence

Copy the desired PIV data text files (.txt) into the **PIVoriginal** folder found in **Data.** Other file types such as .dat can be used but will require slight modifications of the script. The velocity files should be n×4 matrices (x position, y position, x velocity, y velocity).

To make masks, also load the corresponding image sequence into the **'images'** folder found in **'Data'**. Do not worry about frame increment yet. Copy all images based on the start and end frames corresponding to the file number of first and last velocity files. This images will be used to create masks since masks are required for the computation of accurate pressure fields and the calculation of forces at the animal-fluid boundary. Image file type can be .jpg, .tif, or .png, but I recommend using .jpg to save memory and also because any of the color profiles and metadata that usually makes other file type heavy can be safely ignored here.

2. Selecting the true start and end frames corresponding to PIV sequence

Because of the way velocity fields are generally calculated in DaVis, the start and end video frames corresponding to the first and last velocity file do not simply correspond to the frame # divided by the increment between frames used to compute velocity fields. Instead, the corresponding frame number can be calculated as follow:

$$frame\ \# = (Velocity\ file\ \# - 1) \times increment + 1$$

where increment is the time interval between frames used to compute velocity fields. An optional function is provided at the beginning of the master script to perform this calculation (see III.2).

III. Running 'MasterComputePressureWithMask'

Before running the script, make sure to add all the functions and dependencies to the Path.

1. Main paths

This section sets all the paths that will be used to import and export files into different folders. The last line also allows the user to set the image file type to ensure that the remaining sections can import images properly can communicate with various functions. The default is '.jpg'.

2. Calculate first and last image frame (Optional)

This short optional function allows the user to calculate the first and last frame # of the image sequence corresponding exactly to the first and last velocity files of interest (see details in II.2).

3. Select images based on the increment used for PIV data

This section selects only the images of interest, based on the original increment used to compute the velocity fields. Keep in mind that the unnecessary/unused images are removed from the 'images' folder to save space. The duration of this step is proportional to the number of images loaded in the 'images' folder. Heavier files like .tif also increase the duration of this step. Images are also automatically renamed as B#####, starting at B00001.

4. Rotate images if necessary (Optional)

If for any reasons your images are not oriented with the animal swimming up, like those used to compute velocity fields, this optional function can easily rotate all the frames. To save time, it is best to orient all the original frames outside Matlab. WARNING: Rotating images using Microsoft's rotate tool straight from the folder where all the images are found does NOT work. Also, keep in mind that this function only rotates images. It cannot be used to rotate velocity fields in the event that those are not oriented in such a way that the animal is swimming up.
This option is disabled by default to prevent accidentally running it. It is enabled if opts.rotateSequence = 1.

5. Clean original PIV and export velocity data in the proper format

Because .csv or .dat velocity files produced by PIV software (e.g., DaVis, PIVlab) contain column headers that prevent the pressure script from properly loading velocity data, the original velocity files need to be stripped of their headers. In this step, the original velocity files stored in the **'PIVoriginal'** folder are stripped of their column headers, renamed, and renumbered like the image files, and are converted to .csv before being stored in the **'PIVclean'** folder. Cleaned files are set to .csv by default, but if preferred, you can export cleaned files as .dat. However, the cleaned .csv velocity files are used to set the scale when making the outlines, so if you prefer to work with .dat files, you will have to make slight changes to the file extension and to how the velocity data is read in the corresponding section (see section III.10).

6. Options for BW masks

As discussed above, pressure fields around swimming animals can only be computed accurately if the animal is masked. The method used in this script primarily relies on the difference in brightness and contrast between the animal and the background. This works for both traditional and brightfield PIV. The subject is extracted from the background through a series of thresholding steps that generally yield consistent results that require minimal manual post-processing (see fig. 2).

Several options can be set:

- When opts.maskImage is set to 1, the user can interactively select only the area of interest that should be used to make masks. In some cases, reflections on the walls of the tank or other artifacts can cause the thresholding steps to keep those areas rather than the animal. Given that the camera is not moving, a simple masking of the areas to be excluded can be performed directly from the thresholding function 'makeMask3' (see next step, III.7). Simple masking through **'makeMask3'** has the advantage of being repeatable and requires little to no change from one video analysis to the next.
- When opts.checkMasks = 1, the user can check the quality of the masks as they get exported. This has the advantage of giving a good idea of whether some masks will require manual post-processing in future steps. Enabling this setting does not increase processing time significantly.
- When opts.plotComparison = 1, a comparison of the results produced by the different thresholding steps is shown in a 2×2 subplot to facilitate the fine-tuning of the corresponding parameters in 'makeMask3' (see fig. 2). This option is used for testing purposes and is disabled automatically when all the masks are being exported
- opts.inverseMask is used to inverse the colors of images to make white areas black and vice-versa. It must be set to 1 if the subject is dark on a white background (brightfield PIV) and must remain 0 otherwise.

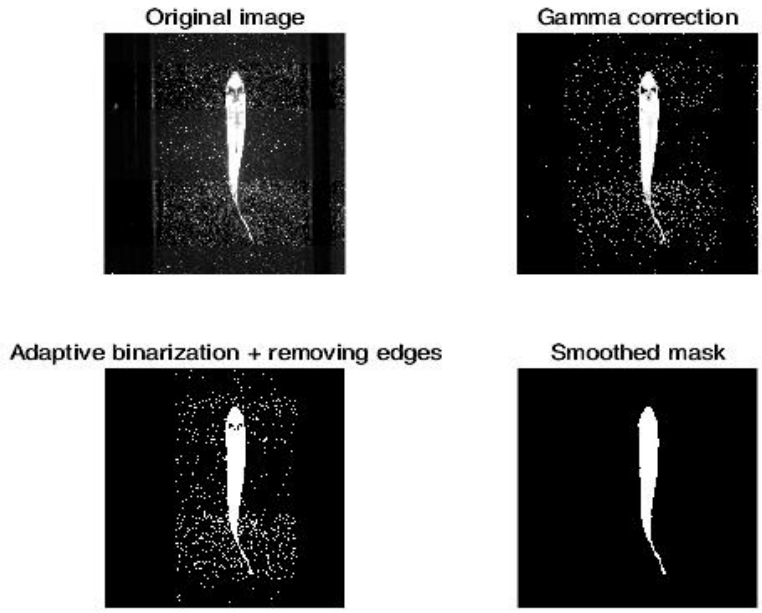

Figure 2. Figures produced during the following testing step that displays the outcomes of the different layers of thresholding. Note the brighter horizontal bands allowing sections of the snout and the tails to be better separated from the background to produce more accurate masks (see details in the following step). The walls of the swim tunnel are masked in the third thresholding step.

7. Test mask and image processing options

It is very likely that the default settings will not produce the desired results. To change the values of the thresholding parameters that control the quality of the masks, open **'makeMask3'** found in the **'ScritsAndFunctions'** folder.

Following are the parameters that you may have to tinker with to achieve the desired results:

- Enhance contrast and preserve edges (these values depend on the original contrast in your images)
  edgeThreshold = 0.6; (0.6 seems to work in most cases)
  amount = 0.5; (the closer to 1, the higher the contrast)

- Vertical correction (case dependent)
  I(:,200:500) = localcontrast(I(:,200:500),edgeThreshold, amount); (excellent to enhance contrast and preserve edges)
  - Example: This increases the contrast in a vertical strip of the image defined here between 200 and 500 pixels (from the left). Increasing the contrast locally can help detach darker areas of the animal from the background. Vertical correction is rarely needed.

- Horizontal correction (also case dependent)
  I(250:650,:) = localcontrast(I(250:650,:),edgeThreshold, amount);
  I(1250:1800,:) = localcontrast(I(1250:1800,:),edgeThreshold, amount);
  - Example: It performs the same contrast correction as the vertical corrections, but along horizontal strips. It is particularly useful for elongated animals like fish for which only small areas near the tail or fins may need to be lightened. Using horizontal strips has the advantage of avoiding increasing noise along the entire body of the animal. Two strips are generally used near the snout and tail for fishes, but if not necessary, these lines can be muted. In this example (see fig. 2), two horizontal strips of the image are defined between 250 and 650 pixels (from the top) and 1250 and 1800 pixels (from the top).

- Adjust contrast and gamma correction on original image
  I2 = imadjust(I,[0.37 1],[],0.2);
  - Example: This is a very important function. The first value represents the minimum value of the original image that will be mapped as white in the final BW mask. The closer to 0, the more darker values will be mapped as white, thus resulting in a very noisy mask. A value closer to 1 decreases noise but may omit entire sections of the animal. Although very case dependent, a common range for the first parameter is 0.25-0.40. The second value should stay at 1, but if the image is slightly overexposed, it can be decreased a bit (but must remain greater than the first value). The third value is a gamma correction. If gamma is less than 1, then imadjust weights the mapping toward higher (brighter) output values. Standard values range between 0.2 and 0.4.

- Mask unwanted horizontal and/or vertical sections of the images
  - It is the equivalent of the 'opts.maskImage' function described in section III.6 but set manually. It offers similar control, but with more repeatability. These sections can be muted to speed the processing of the mask if no masking is needed.

Horizontal (replaces original pixels by black pixels)
BW2(1:500,:) = 0; for pixel row 1 to 500 (from left side of the image)
BW2(size(BW2,2)-700:size(BW2,2),:) = 0; for pixel row starting on right edge of frame to 700 pixels from right edge.

Vertical (replaces original pixels by black pixels)
BW2(:,1:600) = 0; for pixel column 1 to 600 (from left side of the image)
BW2(:,size(BW2,1)-400:size(BW2,1)) = 0; for pixel column starting on right edge of frame to 400 pixels from right edge.

No other parameters should need to be changed other than the ones listed above. Feel free to experiment with different combinations of thresholding parameters or better image brightness and contrast options.

8. Export all the BW masks

If the test mask is satisfactory, you can proceed to exporting all the BW masks into the **'masksBW'** folder. Technically, BW masks are not directly used in the computation of pressure fields and forces, and have no use when plotting velocity vectors, or vorticity. They only serve as the basis to making outlines, the latter being used in any subsequent computations and plots. However, BW masks provide a quick and easy way to visualize and identify potential bad masks without requiring the further use of Matlab.

9. Options for outlines

Good quality outlines are necessary to compute accurate pressure fields. Therefore, a few parameters must be set to ensure that the outlines produced from the BW masks are smooth and fall outside of the boundary layer (see Dabiri et al., 2014 for details about the effect of the boundary layer on pressure calculations).

Several options can be set:

- opts.smooth allows the user to use one of four possible smoothing options to smooth the edges of the outlines. If set to 0, no smoothing is applied, 1 uses moving average, 2 uses loess smoothing, and 3 uses Savitzky-Golay smoothing. By default, option 3 is used and produces very good results.
- opts.smoothn sets the value for the smoothing parameter defined above. See additional details in the 'mask2outline' function found in 'ScriptsAndFunctions'. By default, it is set to 150 for option 3.

- If opts.visualizeExport is set to 1, each outline can been seen as all the outlines get exported. Set to 0 if not needed or to speed up the export.
- PixelDilation is a very important parameter! As described by Dabiri et al. (2014), accurate pressure field computations can be affected by the boundary layer. It is recommended to dilate (widen) the outlines by a few pixels to fall outside of the boundary layer. Keep in mind that the value of this dilation (in pixel) also depends on the original width of the interrogation window size. The larger the interrogation windows, the greater the required dilation. For example, 2048×2048px images containing a fish occupying ¾ of the frame and yielding 128×128 vectors per frame require a dilation factor of 10-15 pixels.

10. Set scale from PIV data to scale outlines

In this step, we set the scale used in the creation of outlines from the original, unscaled BW masks exported in step III.8. This scale is set automatically from one velocity file assuming that velocity vectors are originating from the center of the interrogation windows. Outlines are scaled in the same unit as that of the x and y position stored in velocity files. If using DaVis, this method is appropriate because regardless of any initial masking during the computation of velocity fields, vector fields are generated across the entire image. This means that the minimum and maximum x and y values stored in each velocity file correspond to the edges of the image. Warning: this approach may not work with other PIV software. For example, when masking is applied in PIVlab, only those vectors found within the region of interest will be stored in velocity files. This means that the minimum and maximum x and y values will not be representative of the entire length and width of the frame.

11. Test one outline

In this step, you can visualize the outline produced for one frame to confirm that the smoothing and dilation parameters are set properly (fig. 3).

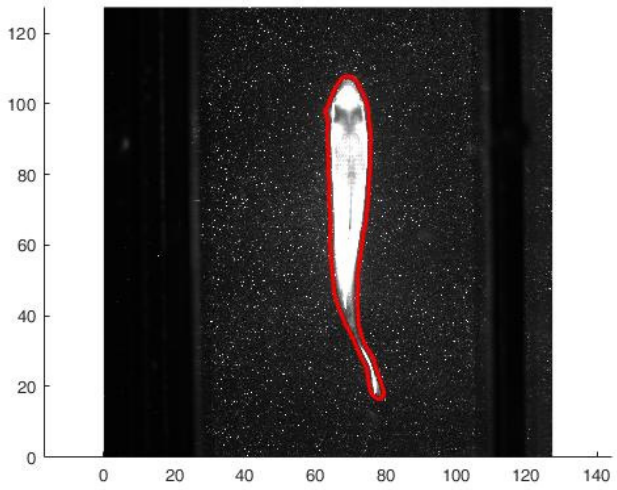

Figure 3. Figure generated during the outline test phase to check that the quality of the outlines is satisfactory.

12. Export all the outlines

In this step, all the outlines are exported in the **'outlines'** folder. If opts.visualizeExport is set to 1, you will see each outline before it is exported.

13. Convert outlines to another unit (Optional)

In the event that you need the outlines to be reported in another unit, this optional function performs a simple conversion. Here the default conversion applies to the metric system.
Similar to other optional sections, it is muted by default by the opts.rescaleOutlines option to avoid its accidental use. Set this option to 1 if a conversion is needed.

14. Repair outlines manually if necessary (Optional)

If the quality of some of the outlines is not satisfactory, individual outlines can be easily modified manually. This first section stores a few useful options:

- When opts.exportOutlines is set to 1 (default) each corrected outline is exported and replaces its older version.
- When opts.checkOutlines is set to 1 (default), the function uses the existing outline instead of making an outline from the BWmask. Generally preferred to save computing time.
- The opts.keepTrackChange option keeps track of which file has been checked. It is generally not needed and is set to 0 by default.
- The opts.repair option allows the user to repair the BW masks before making outlines if necessary. It is not recommended and wastes a lot of time. It is best to keep it at 0 and repair the outline itself instead.
- Similar to section III.9, the opts.smooth and opts.smoothn functions control the amount of additional smoothing applied to the repaired masks before they get exported. It is generally not necessary since step III.9 should have already smoothed the outlines to the desired amount.
- If more dilation is needed, set opts.dilateMask to 1 and enter the corresponding dilation value (in pixel) using opts.dilatePx. This is generally not needed since step III.9 should have already performed the desired amount of dilation.

15. Load files (if running the previous step)

In this step, the outlines and BW masks are loaded in their respective variables for future use in the next section.

16. Start repairing outlines (if running the previous step)

Upon launching this section, a figure will be opened (fig. 4), and outlines can be manually repaired through a GUI. First, if the displayed outline does not need to be repaired, move to the next outline by pressing return. Even if the outline is not repaired, if the smoothing and dilation options were enabled, smoothing and dilation will be applied. If the outline needs to be fixed, press 'a'. You will be prompted to click on the area that needs to be repaired (fig 5A). A close-up view of the area of interest will be displayed (fig 5B). At this point, trace the path that you think is correct by creating as many points as needed using the mouse (fig 5C). Hit return. This will bring you to the

original figure with the repaired outline (fig 5D). If more repairs are needed, press “a” again, and repeat this procedure. Repeat as many times as needed. When the outline is looking good, hit return to move to the next outline. If you need to stop repairing outlines, press ‘s’.
Tip: If for some reason nothing needs to be repaired in the repair step when you are supposed to select points to create a path (fig 5B), simply click once anywhere on the figure and hit enter. To fix the outline, at least two points must be selected. Therefore, if only one point is generated, no corrections is applied.

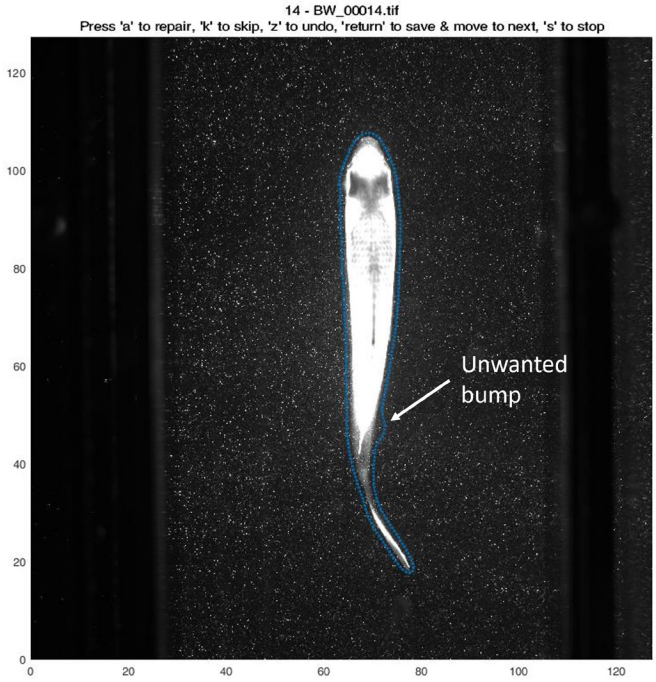

Figure 4. Example of a problematic outline containing an unwanted bump and requiring manual repair.

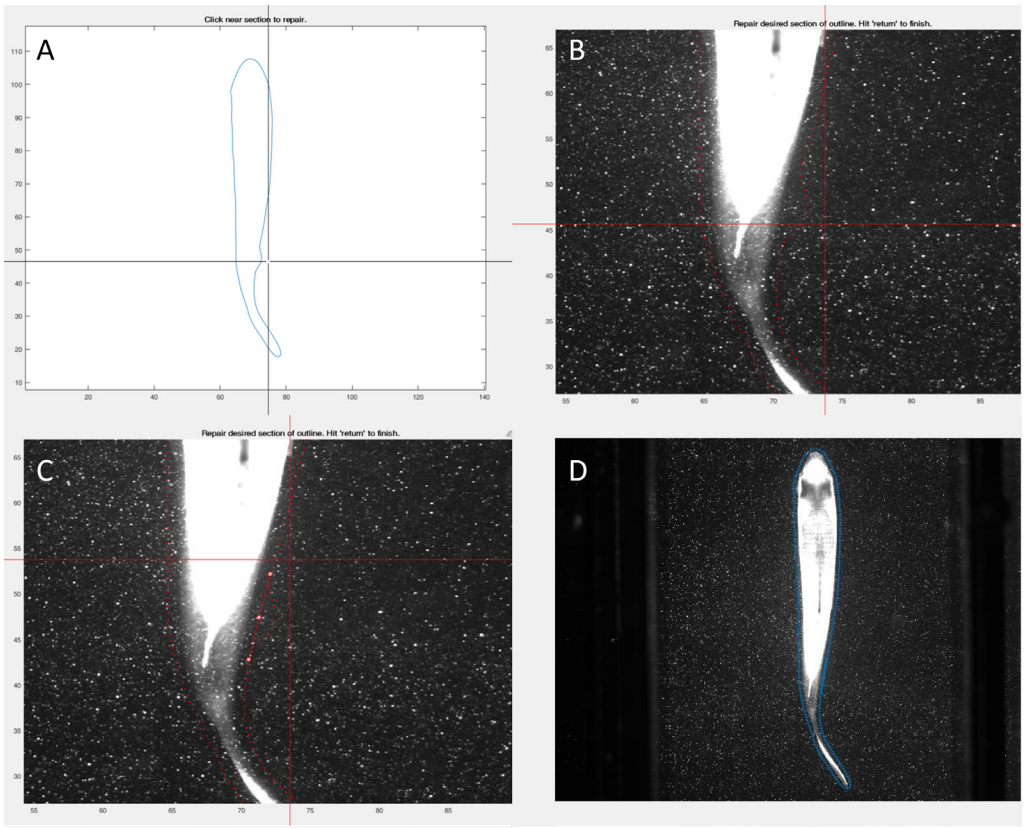

Figure 5. Manual repair of the frame displayed in fig. 4. **(A)** Selecting the general area containing the unwanted outline feature. **(B)** The outline is plotted over the original image to allow for a more intuitive correction of the outline. **(C)** Points corresponding to what the outline should look like are replacing the unwanted outline feature. **(D)** Upon accepting the change, the repaired outline is plotted.

17. Export corrected BW masks using the repaired outlines (Optional)

In this optional step, you can export new BW masks using the repaired outlines. It is set to 0 to avoid running this section accidentally. New masks will replace the original BW masks. This step is not necessary but clean BW masks may be useful for other purposes (e.g., creating an animation). This step is very fast.

18. Calculate pressure fields

Before launching queen2, you will have to set up a few parameters using the **'parameters2'** Matlab file opened in this step. Descriptions of each parameter is provided at the top, but here are the parameters that you will have to pay attention to:

- First: file # of the first velocity file.
- Last: file # of the last velocity file.
- deltaT: This corresponds to the time increment between frames used in the original velocity computations.

- rho: You should not have to change the value for fluid density often, but it must be set properly to obtain accurate pressure fields. It is set by default to 1020.7 kg m^{-3} (seawater at 30ppt and 21°C).
-

19. Run queen2

If everything was set properly in 'parameters2', a GUI will pop up. Press **'Compute Pressure Fields'** to launch the computation (fig. 6). For more details, check the **'Read me'** file located in the **'queen2src'** folder. I changed the original colormap to better visualize the pressure fields. This new colormap uses 'Custom Colormap' by Víctor Martínez-Cagigal (available at https://www.mathworks.com/matlabcentral/fileexchange/69470-custom-colormap?s_tid=srchtitle_customcolormap_1).

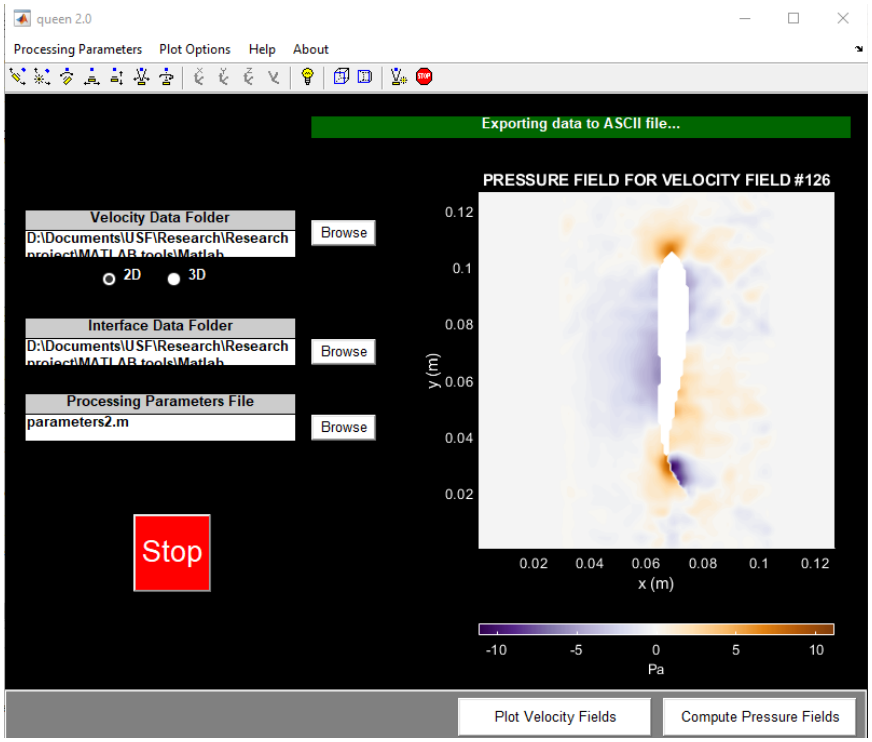

Figure 6. Representative view of the queen2 GUI during the computation of pressure fields. Note that the colormap used here is not the colormap originally loaded in queen2.

20. Move pressure files from the PIVclean folder to the pressure folder

Pressure files (in .csv format) are automatically exported in the 'PIVclean' folder. This section moves all the pressure files to the **'pressure'** folder.

21. Compute forces along the body

In this section, the forces at the animal-fluid boundary are calculated from the pressure fields at points distributed along the outline of the animal.

First, set the following options:

- opts.npoints: set the number of points along the outline to calculate and plot force vectors. It is set to 150 by default.
- opts.forceScale is used to scale the length of the vectors when plotting for aesthetic purposes. This parameter does not affect the magnitude of the forces being computed.
- opts.increment sets the force calculation increment (from one frame to the next). It is preferrable to keep it at 1.
- opts.pressureDeltaT corresponds to the time increment between frames used in the original velocity computations.
- The opts.useRawImage option allows to plot the raw image or the mask (from outline). Set to 0 to plot the outline in black or set to 1 to use the raw image instead.
- When opts.exportFigures is set to 1, all the force figures are exported to the **'forces'** folder.
- To compute forces and later the Froude efficiency, set opts.swimmingSpeed to the corresponding average swimming speed of the animal (in m s^{-1}).

In this section, you will test one frame to ensure that all the parameters are set to your liking. By default, this section will first plot the outline overlaid on the original image, and the pressure fields around the fish for this frame (fig. 7). Then different options can be used for the **'forcesAlongBody'** function to either plot axial forces, lateral forces, vector forces, or a subplot of the three previous options (fig. 8).

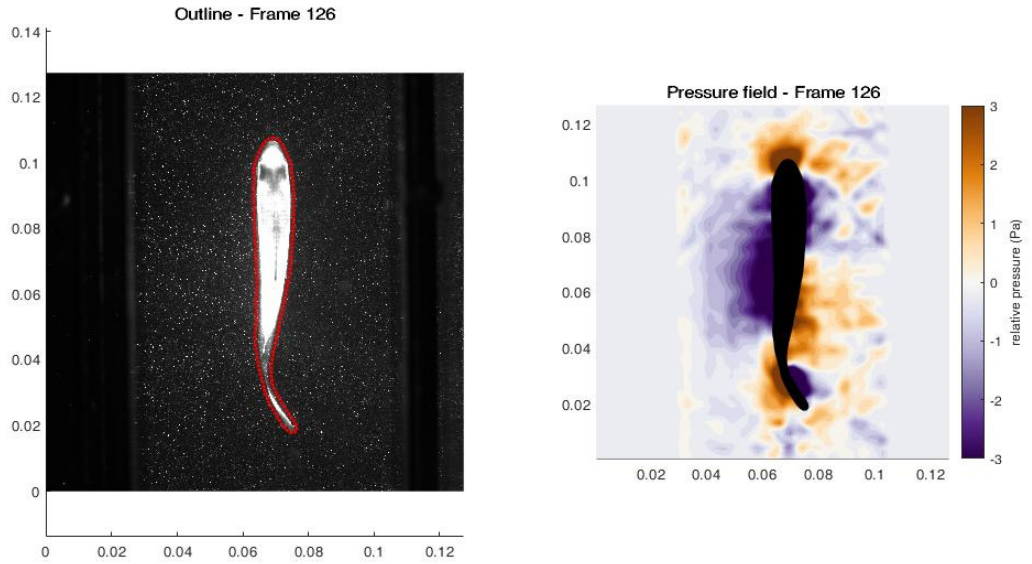

Figure 7. Outline and pressure fields for one test frame. This figure can help verify that the forces computed along the body of the animal match the pressure gradients observed for a desired frame.

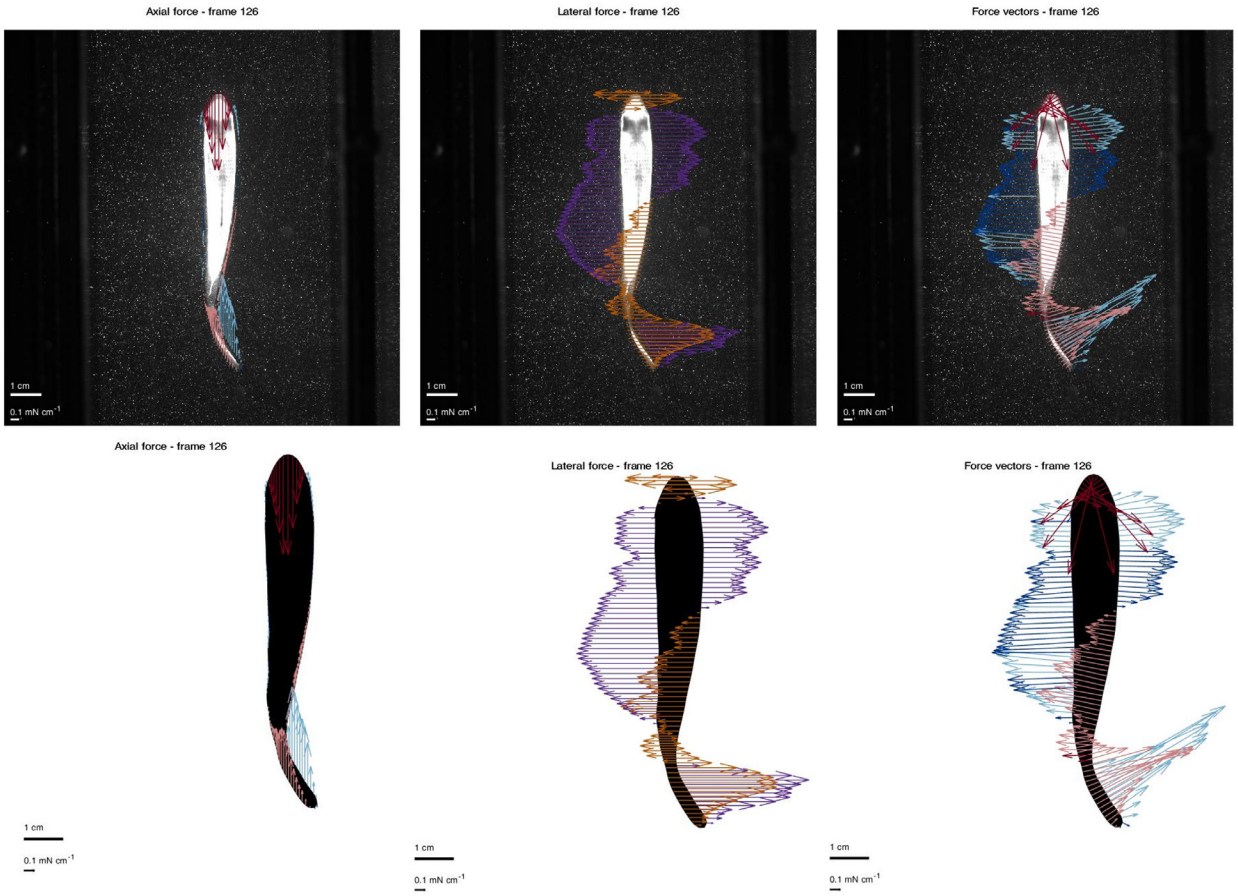

Figure 8. Possible figure outputs of the 'forcesAlongBody' function. The raw image can be used to plot forces (top row) or instead, the outline of the fish (filled with black) can be plotted (bottom row). With this function, the user can plot axial forces, lateral forces, or force vectors in all directions.

22. Export all force figures and data

Run this section to export all the force figures as well as individual .csv files that store force and power data for each frame. The files contain the following data for each point along the outline: x coordinates (m), y coordinates (m), local lateral force component (N m^{-1}), local axial force component (N m^{-1}), local lateral power (W m^{-1}), local axial power (W m^{-1}). An additional file is exported to summarize the following data: frame #, total pull thrust (N m^{-1}), total push thrust (N m^{-1}), total pull drag (N m^{-1}), total push drag (N m^{-1}), net force (N m^{-1}), total axial power (W m^{-1}), total lateral power (W m^{-1}), total axial pull power (W m^{-1}), total axial push power (W m^{-1}), total lateral pull power (W m^{-1}), total lateral push power (W m^{-1}), and Froude efficiency.

23. Plot net force over time

This section allows the user to plot the net force produced by the animal overtime (fig. 9).

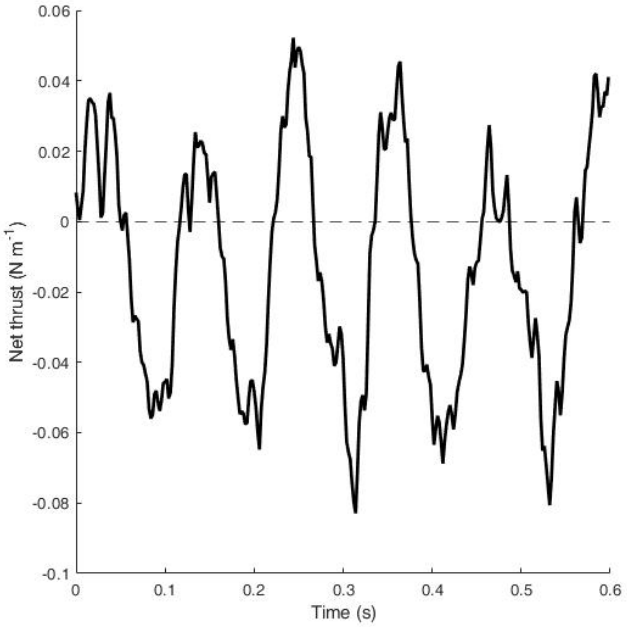

Figure 9. Net thrust over time produced through the computation of forces along the body of a free-swimming fish.

24. Calculate the average Froude efficiency for 1 tail beat

In this section, the average Froude efficiency over one tail beat cycle is calculated. You must first input the frame # where the tail beat starts and the frame # where the tail beat ends. This section returns the Froude efficiency in the command window.