von Karman Institute for Fluid Dynamics
Chaussée de Waterloo, 72
B - 1640 Rhode Saint Genèse - Belgium

Project Report

Implementation of iterative Multigrid and Window Deformation Schemes in the OpenPIV Python Package

T. Käufer

Supervisors:
M. A. Mendez, von Karman Institute
A. Liberzon, Tel Aviv University

December 2019

Chapter 3

Advanced PIV interrogation using the OpenPIV Python package

The implemented advanced interrogation techniques became part of the OpenPIV Python package [8]. One can install it using pip or download it from GitHub here. The Iterative Multigrid and Window Deformation Schemes algorithm uses a Python version of the"smoothn" algorithm from D. Garcia [5]. The easiest way to set up and run a PIV analysis using advanced interrogation techniques is to open the example_windef_run.py file and enter the desired parameter.

3.1 The example_windef_run.py

The example_windef_run.py is a setup file for the PIV interrogation. After modifying the parameters according to one's needs one just runs the script and the results are returned in the specified folder. The available options are:

Data related settings

- **settings.filepath_images:** (type: string)
  Here the file path to the folder that contains the images must be added. Note: If one uses the absolute file path on a windows system ":\" can result in an error. This can be solved by adding a second backslash like this ":\\".

- **settings.save_path:** (type: string)
  Here the file path must be inserted at which a folder with the results should be created. Note: Be aware about the absolute file path issue mentioned above.

- **settings.save_folder_suffix:** (type: string)
  Here a suffix can be added to customize the name of the results folder. The save folder is named like:
  "OpenPIV_results_"+*window size of the last pass*+"_"+*settings.save_folder_suffix*

- **settings.frame_pattern_a and settings.frame_pattern_b:** (type: string)
  These parameters should contain the pattern of the image's names. An Example: Assuming 200 images structured as 100 double frames and the images names to be like A000a-A099a and A000b-A099b. In this case one would like to correlate the image A000a with the image A000b and so on until all the images are evaluated. This is automatized by using a pattern. For this example, the **settings.frame_pattern_a** would look like **A*a** and **settings.frame_pattern_b** would look like **A*b**. The asterisk is replacing the frame number.

Region Of Interest

- **settings.ROI:** (type: string or tuple with four int entries)
  This option allows to select a certain Region Of Interest of the images. The **ROI** is a square defined by the pixel coordinates like this **(xmin, xmax, ymin, ymax)** with the image origin in the upper left corner of and the x-axis going to the right and the y-axis going downwards. An example ROI could look like this (50, 300, 50, 300). Be aware that the ROI must not exceed the size of the image. In case one would like to evaluate the whole image, one can replace the tuple with the string 'full'.

Image preprocessing

- **settings.dynamic_masking_method:** (type: string) This option enables the image preprocessing. One can choose between 'None', 'edges' and 'intensity' If 'None' is selected no preprocessing is done. If 'edges' is selected a filter is applied which blurs out edges in the image. If 'intensity' is selected, larger image areas which intensity is above a threshold defined by **settings.dynamic_masking_filter_size:** are masked out. A description of this tool is provided here.

- **settings.dynamic_masking_filter_size:** (type: int) This parameter defines the size of the filter used for the image prepossessing.

- **settings.dynamic_masking_threshold:** (type: float) Here one can insert the threshold for the image prepossessing.

Processing parameters

- **settings.correlation_method:** (type: string) One can choose between to different cross-correlation options. The options are 'circular' and 'linear'. If 'circular' is select circular cross-correlation is done which is faster but slightly less accurate. If 'linear' is the displacement evaluation is done using linear cross-correlation which is slower.

- **setting.iterations:** (type: int)
  This parameter determines the number of PIV-iterations. The minimum is one iteration. Advanced interrogation techniques require at least two iterations.

- **settings.windowsizes:** (type: tuple of int)
  (settings.windowsizes) allows to define the window size for each PIV-iteration. The tuple needs at least as many entries as iterations performed. The first iteration of the PIV evaluation is done using the window size with the index 0 and so on. Only the first pass is affected by the one-quarter rule and the window size of the following passes can be reduced to increase the dynamic range and the spatial resolution. The window size should be a value with base two. Example: (settings.windowsizes)=(64,32,16). Observe that if the tuple introduced is longer than the number of iterations, the last values will be ignored.

- **settings.overlap:** (type: tuple of int)
  settings.overlap allows one to define the overlap of the interrogation windows for each PIV-iteration. The tuple needs at least as many entries as iterations performed. In general, an overlap of half the interrogation window size is a solid choice for most cases. They overlap should be a value with base two. Example: (settings.overlap)=(32,16,8).

- **settings.subpixel_method:** (type: string)
  This option allows one to select a fitting function for the subpixel estimation of the correlation peak. One can choose between 'gaussian', 'centroid' and 'parabolic'. By default, 'gaussian' is used.

- **settings.interpolation_order:** (type: int)
  settings.interpolation_order defines the order of interpolation used for the image deformation. It should be between three and five. Only used if more than one iteration is executed.

- **settings.scaling_factor:** (type: int)
  The scaling factor is used to transform the displacement from $\frac{pixel}{frame}$ to $\frac{meter}{frame}$. It describes the relationship between the measurement plane and the camera sensor. If no information is available, it is better to set this value to 1 and then leave the measurement scaling to the post-processing stage.

- **settings.dt:** (type: int)
  This parameter defines the time between the frames. It transforms the displacement from $\frac{pixel}{frame}$ to $\frac{pixel}{second}$. In combination with the scaling factor, this results in a velocity. If no information is available, as for the previous parameter, it is worth setting it to 1.

Signal to noise ratio options (only for the last pass)

- **setting.extract_sig2noise:** (type: bool)
  This setting lets one choose to extract the signal to noise ratio and export it. Observe that the signal to noise ratio is only calculated for the last pass. It must be enabled to do the signal to noise ratio validation. If this is set to false, the signal to noise ratio column in the output is filled with NaNs.

- **setting.sig2noise_method:** (type: string)
  Here one can choose between the method 'peak2peak' and 'peak2mean' to determine the signal to noise ratio. 'peak2peak' calculates the signal to noise ratio by dividing the correlation value of the largest peak by the correlation value of the second largest peak. 'peak2mean' calculates the signal to noise ratio by dividing the largest peak by the averaged correlation value of the interrogation window.

- **settings.sig2noise_mask:** (type: int)
  This parameter defines the size of the mask around the first peak. It only affects the calculation of the signal to noise ratio if the option 'peak2peak' is chosen. 'peak2peak' calculates the signal to noise ratio by dividing the value of the largest correlation peak with the value of the second largest peak. In the case of very noisy cross-correlation maps, this function avoids that the second correlation peak is too close the first. This is done by using a circled mask centered in the first peak. This parameter is the radius of such a mask. By default, two is a sufficient value.

Vector validation options

- **setting.validation_first_pass** (type: bool)
  In the multi-pass approach, the first pass is always validated before proceeding to the following ones. If Singlepass is chosen, the validation can be disabled by setting this parameter to false.

  Validation by minimal and maximal displacement

  - **settings.MinMax_U_disp and settings.MinMax_U_disp:** (type: tuple of float)
    This option allows one to set a limit for the u and v displacement. The values have to be inserted into a tuple like this (min, max). Displacement vectors that exceed these limits will be marked as invalid. In case this kind of validation is not desired, a very large value can be inserted. Important: the input for this validation is in $\frac{pixel}{frame}$.

  Validation by standard deviation

  - **settings.std_threshold:** (type: float)
    This parameter sets the threshold for the validation by the global standard deviation. Vectors are discarded if they differ from the global mean by a certain value, in either of the components, which is larger than the specified number of standard deviations. A sufficient value has to be chosen according to the circumstances. Also, this validation can be disabled by entering very large thresholds.

  Validation by local median

  - **settings.median_threshold:** (type: float)
    This parameter sets the threshold for the validation by the local median. For that, the local median of the u and v displacement is calculated independently. In case the difference between the actual value and its local median exceeds the threshold, the vector is masked as invalid. This validation is similar to the popular median test but has no normalization. Like the others, it can be disabled by choosing a large value.

  Validation by signal to noise ratio

  - **settings.do_sig2noise_validation:** (type: bool)
    This enables or disables the validation by the signal to noise ratio by setting it to True or False.

It is only available for the last pass and **setting.extract_sig2noise** has to be enabled.

- **settings.sig2noise_threshold:** (type: float)
  Sets the threshold for the signal to noise ratio. Displacement vectors that exceed this threshold are marked as invalid and removed.

Outlier replacement or Smoothing options

- **settings.replace_vectors:** (type: bool)
  This option enables or disables the outlier replacement by setting it to True or False. This is only valid for the last pass.

- **settings.smoothn:** (type: bool)
  This option enables or disables the smoothing of the velocity fields. This option uses the famous smoothing function from Garcia, also used in PIVlab. It can be of interest when spatial derivatives will be calculated on the instantaneous velocity fields.

- **settings.smoothn_p:** (type: float)
  This is the smoothing parameter from Garcia's function. Low values produce little smoothing, high values produce strong smoothing.

- **settings.filter_method:** (type: string)
  This parameter stipulates the method used to replace the outlier vectors. One can choose between 'localmean', 'disk' and 'distance'. 'localmean' replaces the invalid vector by the mean of a local square area around the invalid vector. 'disk' replaces the invalid vector by the mean of a local circle around the invalid vector. 'distance' uses also a circular area around the invalid vector but weights the values according to their distance from the center.

- **settings.max_filter_iteration:** (type: int)
  settings.max_filter_iteration defines the maximum number of filter iterations. This kind of filter, documented in the OpenPIV package, is important when a cluster of outliers is produced. In this case, the replacement starts from the boundaries of the field and proceed towards the center using a number of iterations defined by this parameter. If this value is set too high, the value used to replace the invalid vector can significantly diverge from the actual displacement and it might be worth considering different parameters for the analysis.

- **settings.filter_kernel_size** (type: int) Here one can define the size of the area used to calculate the vector replacement. Default: 2

Output options

- **settings.save_plot:** (type: bool) Decide whether one wants to save the plotted vector fields by choosing between True and False. They are stored in the same folder as the .txt files containing the results.
- **settings.show_plot:** (type: bool) Decide whether one wants to see the plotted vector fields by choosing between True and False. Depending on the interrogation setting the vector field may be instantly replaced by the following vector field.
- **settings.scale_plot:** Defines the arrow size of the plotted vector field. Larger values result in smaller arrows.

The code of the script is shown below:

```
from openpiv import windef


settings = windef.Settings()


'Data related settings'
# Folder with the images to process
settings.filepath_images = r'C:\Users\Theo\Desktop\VKI\Validaton\PIV_challenge_2003_A'
# Folder for the outputs
settings.save_path = r'C:\Users\Theo\Desktop\VKI\Validaton\Vectorfields_2003A'
# Root name of the output Folder for Result Files
settings.save_folder_suffix = 'test'
# Format and Image Sequence
settings.frame_pattern_a = 'A*a.tif'
settings.frame_pattern_b = 'A*b.tif'

'Region of interest'
# (50,300,50,300) #Region of interest: (xmin,xmax,ymin,ymax) or 'full' for full image
settings.ROI = 'full'

'Image preprocessing'
# 'None' for no masking, 'edges' for edges masking, 'intensity' for intensity masking
# WARNING: This part is under development so better not to use MASKS
settings.dynamic_masking_method = 'None'
settings.dynamic_masking_threshold = 0.005
settings.dynamic_masking_filter_size = 7
```

```
'Processing Parameters'
settings.correlation_method='circular'  # 'circular' or 'linear'
settings.iterations = 2  # select the number of PIV passes
# add the interroagtion window size for each pass.
# For the moment, it should be a power of 2
settings.windowsizes = (64, 32, 16) # if longer than n iteration the rest is ignored
# The overlap of the interroagtion window for each pass.
settings.overlap = (32, 16, 8) # This is 50% overlap
# Has to be a value with base two. In general window size/2 is a good choice.
# methode used for subpixel interpolation: 'gaussian','centroid','parabolic'
settings.subpixel_method = 'gaussian'
# order of the image interpolation for the window deformation
settings.interpolation_order = 3
settings.scaling_factor = 1  # scaling factor pixel/meter
settings.dt = 1  # time between to frames (in seconds)
'Signal to noise ratio options (only for the last pass)'
# It is possible to decide if the S/N should be computed (for the last pass) or not
settings.extract_sig2noise = False  # 'True' or 'False' (only for the last pass)
# method used to calculate the signal to noise ratio 'peak2peak' or 'peak2mean'
settings.sig2noise_method = 'peak2peak'
# select the width of the mask to mask out pixels next to the main peak
settings.sig2noise_mask = 2
# If extract_sig2noise==False the values in the signal to noise ratio
# output column are set to NaN
'vector validation options'
# choose wether you want to do validation of the first pass: True or False
settings.validation_first_pass = True
# only effecting the first pass of the interrogation the following passes
# in the multipass will be validated
'Validation Parameters'
# The validation is done at each iteration based on three filters.
# The first filter is based on the min/max ranges. Observe that these values are defined in
# terms of minimum and maximum displacement in pixel/frames.
settings.MinMax_U_disp = (-10, 5)
settings.MinMax_V_disp = (-5, 5)
# The second filter is based on the global STD threshold
settings.std_threshold = 7  # threshold of the std validation
# The third filter is the median test (not normalized at the moment)
settings.median_threshold = 5  # threshold of the median validation
# On the last iteration, an additional validation can be done based on the S/N.
settings.median_size=1 #defines the size of the local median
'Validation based on the signal to noise ratio'
# Note: only available when extract_sig2noise==True and only for the last
# pass of the interrogation
# Enable the signal to noise ratio validation. Options: True or False
settings.do_sig2noise_validation = False # This is time consuming
# minmum signal to noise ratio that is need for a valid vector
settings.sig2noise_threshold = 1.2
```

```
'Outlier replacement or Smoothing options'
# Replacment options for vectors which are masked as invalid by the validation
settings.replace_vectors = True # Enable the replacment. Chosse: True or False
settings.smoothn=False #Enables smoothing of the displacemenet field
settings.smoothn_p=0.1 # This is a smoothing parameter
# select a method to replace the outliers: 'localmean', 'disk', 'distance'
settings.filter_method = 'localmean'
# maximum iterations performed to replace the outliers
settings.max_filter_iteration = 4
settings.filter_kernel_size = 2  # kernel size for the localmean method
'Output options'
# Select if you want to save the plotted vectorfield: True or False
settings.save_plot = False
# Choose wether you want to see the vectorfield or not :True or False
settings.show_plot = True
settings.scale_plot = 500  # select a value to scale the quiver plot of the vectorfield
# run the script with the given settings

windef.piv(settings)
```