PIV Image Generator - Technical Manual

Luís Mendes

May 11, 2020

1 Introduction

The PIV Image generator tool is a tool for generating synthetic Particle Image Velocimetry (PIV) images of fluid flows with the purpose of evaluting the accuracy of PIV and Optical Flow methods, or other methods that are based on analysis of tracer based images, regarding velocimetry applications in fluid mechanics.

The PIV Image generator tool is implemented in MATLAB language, but it may be ported to other languages, like Python. If a port is released, it will also be available under the same public git repository. The code is licensed under GNU General Public License v2.0.

The code is available at https://git.qoto.org/CoreRasurae/piv-image-generator.

This document consitutes the technical manual of the above identified software, detailing some implementation decisions, thus complementing the source code.

2 Technical manual

The PIV image generator tool is composed by the application code and the supporting library code. The application code is composed by files: generatePIVImagesWithAllParametersCombinations.m; exampleGenerateTestImageMain.m and exampleAllTestImagesMain.m. Where exampleGenerateTestImageMain.m, exampleAllTestImagesMain.m are the user files and generatePIVImagesWithAllParametersCombinations.m is the application. All other files are the library files. This document is mainly focused on the library files and implementation. The file generatePIVImages.m is the library entry file through generatePIVImages(...) function.

The images and validation data are generated according to the flow diagram presented in Figure 1.

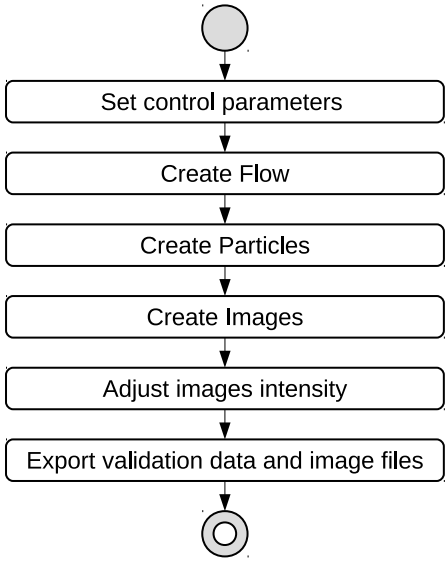

Figure 1: Flowchart diagram of PIV image generator tool

The initialization step "Set control parameters" amounts to the user creating the following structures and parameters:

- sizeX - defines the width in pixels for the generated image, which must be an exact multiple of the IA size
- sizeY - defines the height in pixels for the generated image, which must be an exact multiple of the IA size
- flowParameters - defines the flow type and relative velocity
- pivParameters - defines PIV specific configuration parameters

In addition, the initialization step also derives additional values and parameters from the input parameters and built-in constants, as well as definitions. The implementation files involved are generatePIVImages.m, generatePIVImagesWithAllParametersCombinations.m and either exampleGenerateTestImageMain.m, or exampleAllTestImagesMain.m. Files exampleGenerateTestImageMain.m, or exampleAllTestImagesMain.m constitute two different user entry points: the first for creating a single image pair and the second for creating multiple image pairs with different control parameter combinations.

The flowParameters structure fields are shown in Figure 2 and are detailed as follows:

- flowParameters.maxVelocity is a builtin value in the code that defines a target maximum velocity of 1000.0 mm/s;

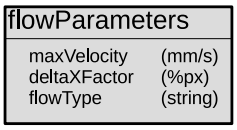

Figure 2: Flow parameters input configuration structure fields

- flowParameters.deltaXFactor is set by the user and defines the displacement ratio relative to the IA size defined in pivParameters structure. In the end this deltaXFactor defines the approximate maximum displacement D_{max} that will occur in the generated image. If IA size is $16px \times 16px$ and $deltaXFactor = 0.25$, then maximum displacement will be $4px$ both in x and y direction.

- flowParameters.flowType is a string that identifies the desired flow field to employ during image generation. Valid values for the current version are: 'uniform' for uniform flow; 'parabolic' for parabolic flow; 'rankine_vortex' for Rankine vortex; 'rk_uniform' for Rankine vortex with superimposed uniform flow; 'stagnation' for inviscid stagnation point flow field; 'shear' for horizontal shear flow; 'shear_22d3' for shear flow rotated by 22.3 degrees, 'shear_45d0' for shear flow rotated by 45.0 degrees and 'decaying_vortex' for Decaying (Lamb-Oseen like) vortex.

The pivParameters structure fields are identified in Figure 3 and are detailed as follows:

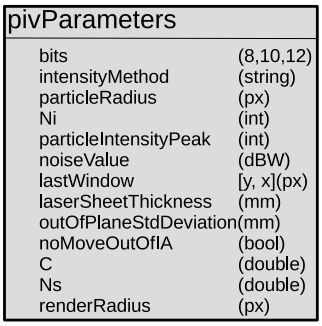

Figure 3: PIV parameters input configuration structure fields

- pivParameters.bits defines the image pixel bit depth, that is, the gray intensity levels for the image, having typical configuration values of 8, 10 or 12 bits. Values above 8 trigger the generation of 16-bit per pixel images.

- pivParameters.intensityMethod defines the tool behavior for limiting the final intensity levels to the desired pixel bit depth. Allowed values are

'clip' and 'normalize'. The 'clip' option, as the name states will clip values above the maximum intensity level for the set bit-depth to the maximum intensity level permitted, while the 'normalize' option will apply a scale factor so that the scaled maximum value matches the maximum intensity level allowed for the pixel bit depth.

- pivParameters.particleRadius defines the typical tracer particle radius in px.
- pivParameters.Ni defines the effective particle image density, i.e., the exact number of visible particles present in each IA for the first image.
- pivParameters.particleIntensityPeak is set by the application code to a predefined expression, having the default expression value of

$$150.0 * maxValue/255.0,$$

  where the value 150.0 can be adjusted in the range $0 - 255$ and maxValue scales the value in accordance to the pixel bit depth set in pivParameters.bits.
- pivParameters.noiseValue defines the WGIN value in dBW to be applied to the generated images.
- pivParameters.lastWindow is an array that defines the dimensions of the PIV IAs at the last step of the PIV computation, when using multiple steps with different IA sizes, but having no IAs overlap. The parameter is defined as the array $[< IAy > < IAx >]$, where $< IAy >$ denotes the y size of the IAs and $< IAx >$ denotes the x size of the IAs, both sizes are in px.
- pivParameters.laserSheetThickness specifies the thickness of the light sheet in millimeters, thus defining the volume of the IA in conjunction with pivParameters.lastWindow. Despite both parameters being specified in different units, it is possible to convert between them according using the mmPerPixel unit conversion factor.
- pivParameters.outOfPlaneStdDeviation translates the out of plane movement, i.e. movement in the z direction, due to turbulence in the form of the standard deviation of the particle position relative to the center of the light sheet.
- pivParameters.noMoveOutOfIA is a boolean value that defines if the particles are allowed to move out of the IA from the first to the second image. If set to true, the particle placement will be limited to the inner region of the IA given the maximum displacement, otherwise the particles can be placed anywhere in the IA, resulting in more realistic PIV images and behavior.

- pivParameters.C is the concentration of particles in the laser sheet and is derived from user provided values. The user doesn't need to fill this value.
- pivParameters.Ns is the source density and is derived from the user provided values. The user doesn't need to fill this value.
- pivParameters.renderRadius is the limit radius, in pixels, around the center of the particle where the 2D gaussian is to be rendered when creating the PIV images.

The first processing step "Create flow" is built around a static factory method that instantiates the requested flow object. The factory gives some flexibility and provides an easy way to extend the tool with new flow types, since the changes only require the definition of a new flow type class. A flow object defines the displacement across the flow field, for a given time between frames, according to the expressions presented in the previous sub-section. Each flow object has a computeDisplacementAtImagePosition(...) method which takes as input the current particle position in the flow field and computes the final position for the particle after the fixed time between frames. The time between frames is set at object creation time, along with the max velocity in px/s. The particle position contains sub-pixel information, thus positions aren't integer values. The object is used at a later stage. This functionality is provided by functionsLib/creatFlow.m, the static factory, and implementation files: functionsLib/ParabolicFlow.m; functionsLib/UniformFlow.m; functionsLib/StagnationFlow.m; functionsLib/RankineVortexFlow.m functionsLib/LambOseenLikeVortexFlow.m; functionsLib/RotatedShearFlow.m and functionsLib/RankineVortexAndUniformFlow.m.

At the second processing stage "Create Particles" the following sub-steps are performed:

- IAs are created with an extra IAs border all around. The IAs border is placed so that additional particles are available to move from the extra border IAs into the relevant outer IAs, in the same way that particles are available to move between inner adjacent IAs.
- Two additional layers, with the same area of the IA, are created at the upper and lower boundaries of the illuminated IA volume, each with thickness $pivParameters.outOfPlaneStdDeviation \times 10$ to simulate real PIV behavior where particles are more or less evenly distributed in the fluid and not just in the illuminated volume, such that particles can move into and out of the illuminated volume. The thickness was determined empirically to serve well the purpose, while keeping acceptable computational complexity. Both the lower and upper layers have a similar number particle concentration per volume, which is given by the expression:

$$\frac{pivParameters.laserSheetThickness}{pivParameters.Ni} \times 10 \times pivParameters.outOfPlaneStdDeviation.$$

This way the particle concentration per unit volume remains approximately constant across all layers, as would be expected in a typical real scenario.

- Particles are individually distributed in each IA for each of the three IA layers, using random uniform distribution both in the x, y and z.
- Particles light intensity for the first image is computed for the initial out of plane (z) position according the expression suggested by [1]:

  $$pivParameters.particleIntensityPeak \times e^{-z^2/(\frac{1}{8} \times pivParameters.laserSheetThickness^2)},$$

  which generates an intensity value according to the position of the particle in the light sheet in the z direction, modeling real PIV behavior.
- Particle out of plane movement is computed for each particle in each layer with a value obtained from a uniform distribution in the range of [0, pivParameters.outOfPlaneStdDeviation].
- Particle light intensity for the final image is computed using the same expression for the first image, but the z position is now updated to reflect the final particle position, that is, the initial position updated by the out of plane movement increment.
- Both a particle map is created for each IA and particle map for all IAs, both with particles from all the three layers containing all the computed information for the particle, i.e., initial position (x,y,z), initial and final intensities and out of plane, z, movement.

This stage is implemented in the file functionsLib/createParticles.m.

The third processing stage "Create Images" is comprised by the following:

- Displacements are computed for each particle initial position, obtaining the final particle position
- Particle is rendered in the initial image using the initial particle intensity and is rendered as a 2D Gaussian using the expression in [1]

  $$I \times e^{-[(x+0.5-x_c)^2+(y+0.5-y_c)^2]/(\frac{1}{8}d^2)},$$

  where I is the particle intensity at the particle center, (x,y) is the particle pixel element being rendered, (x_c,y_c) is the particle center, d is the particle diameter.
- The same particle is rendered for the final image, now at the final position after displacement with the final particle intensity using the same 2D Gaussian as in the initial image.
- The extra IAs border is removed from the generated images.

- WGIN noise is added independently to the images according to the pivParameters.noise value and the pivParameters.bits value.

This processing step is implemented in: functionsLib/createImages.m and functionsLib/renderParticle.m.

The step "Adjust images intensity" limits the generated images intensity levels to the pixel bit-depth set in pivParameters.bits by employing the method selected in pivParameters.intensityMethod, as documented for the initialization step in pivParameters fields descriptions. Functionality is implemented in functionsLib/adjustImagesIntensity.m.

The final processing stage "Export validation data and image files" is described by the following sub-steps:

- Exact optical flow field is computed by finding the displacements at the center of each pixel position
- Estimated PIV flow field is computed by finding the displacements at the center of each IA
- Fi, Fo and Rd PIV quality estimation parameters are computed for each IA, where Fi is the in-plane loss factor, Fo is the out-of-plane loss factor, Rd is the estimated cross-correlation quality. Fi translates to the number of particles that moved outside the IA while remaining in-plane to the illuminated region from the first to the second image, as well as particles that moved from adjacent IAs into the IA under evaluation. Fi is decomposed into FiIns and FiOuts, where FiIns is the number of particles that enter in-plane into the IA in the second image, while the FiOuts is the number of particles that move out of the IA but still in-plane to the illuminated volume. Similarly Fo is decomposed into FoIns and FoOuts, where FoIns is the number of particles that enter into the IA in the second image due to inwards out-of-plane movement, thus becoming visible, and FoOuts is the number of particles that move out of the illuminated plane in the second image, due to an outwards out-of-plane movement. Rd parameter is estimated from the following expression

  $$Rd = Ni \times (1 - FiOuts/Ni) \times (1 - \Delta z/(pivParameters.laserSheetThickness/2.0)),$$

  where Δz is estimated from the inverse cumulative distribution function of a normal distribution defined as $\mathrm{N}(0, pivParameters.outOfPlaneStdDeviation^2)$ and a 95% confidence interval is used. Rd translates to the estimated final number of particles in the IA that is common to both the first and second images.
- MATLAB file is exported with exact optical flow field under the structure named 'exactOpticalFlowDisplacements', the estimated PIV flow field under the structure 'estimatedPIVDisplacements', FiIns, FiOuts, FoIns,

FoOuts, Rds under the structure named 'pivStatistics', along with the input parameters as well as derived parameters in structures named 'flowParameters', 'pivParameters' and 'imageProperties'.

- Image pair is exported to the 'out' folder under a proper path and image file names.

The respective implementation files are functionsLib/exportFlowFields.m, functionsLib/computeEstimatedPIVFlowField.m, functionsLib/computeExactOpticalFlowField.m and functionsLib/computeFiFoAndRd.m.

References

[1] Raffell, M., Willert, C., Scarano, F., Kähler, C., Wereley, S., Kompenhans, J. Particle Image Velocimetry - A practical guide. *Springer* https://doi.org/10.1007/978-3-319-68852-7 (2018).