Fluere for Particle Image Velocimetry: User Manual

Kyle P. Lynch (k.p.lynch@tudelft.nl)
PhD Student, Delft University of Technology

Accompanies Fluere Version 1.3 - January 2011

Fluere is a intuitive, user-friendly software package for performing particle image velocimetry analysis on sets of digital image data. The program is available on multiple platforms and is distributed as open source software available under the terms of the GNU General Public License (GPL). This document provides a complete tutorial on the installation and use of Fluere, and also includes a detailed description and validation of the algorithms used in the processing.

Contents

1 Installation

Fluere may be installed either from precompiled binaries or by compiling the source code. The pre-built binaries provide the easiest way to get started with the software, but compiling from source allows you to customize the software as needed or contribute to development. All binary and source code releases of Fluere are available from http://fluere.kplynch.com.

1.1 Installing from precompiled binaries

Binaries are available for both Windows, Mac OS X, and x86 Linux. The installation of the Windows version is very straightforward: download the binary ZIP file and extract the files to a directory of your choice. Fluere can be launched by clicking on the executable file within the directory. Mac OS X binaries are distributed as DMG disk images. Double clicking on the DMG file will mount the disk image, in which you will see the Fluere app bundle. Drag the app to a location of your choice, typically the Applications folder, and double click to run. Linux packages are also simple to install. Unpack the zip file to a directory of your choice, then run Fluere by using the shell script contained in the folder, NOT the executable itself! This can be done on the command line by typing *sh Fluere.sh*.

1.2 Installing from source

To compile Fluere from source, the following packages are required to be installed on your system:

Qt4 The GUI Toolkit Qt, available at http://qt.nokia.com/. I recommend installing the Qt SDK which includes the Qt Creator editor, Qt Designer for dialogs, and also includes the MinGW compiler suite when installed on Windows.

FFTW The "Fastest Fourier Transform in the West" FFT library, available at http://www.fftw.org/. You can either install this from source (relatively simple for Mac OS X/Linux), or from precompiled binaries (for Windows). The binary version will consist of a ZIP archive containing a number of DLL and header files, which you need to extract to a directory on your computer.

LibTIFF The TIFF image library, available at http://www.remotesensing.org/libtiff/. As with FFTW, you can either install this package from source, or from precompiled binaries. For Windows users, precompiled binaries are available through the GnuWin32 project at http://gnuwin32.sourceforge.net/packages/tiff.htm.

Armadillo The Armadillo C++ linear algebra library, available at http://arma.sourceforge.net/. This package can either be installed from source or from available binaries.

After you have installed each of these packages, you will be able to build Fluere from either the command line or from a development environment such as Qt Creator. First you will need to configure the build environment. In particular, the `Fluere.pro` file will need to be modified to point to the locations of the above installations. The relevant lines to edit will be located at the top of the file. For Mac OS X and Linux, FFTW, LibTIFF, and Armadillo will typically install in `/usr/local/`, and thus the compiler should automatically find the files. On Windows, I typically move the relevant files into an includes directory, and then reference this one directory. After these modifications are made, you should be able to compile Fluere either on the command-line or with the tools in Qt Creator. For more information on how to use the command-line Qt build system, visit http://doc.qt.nokia.com/latest/qmake-manual.html.

Note: On Mac OS X systems, Apple only ships version 4.2.1 of the GCC compiler by default when the development tools are installed. This is an antiquated version which does not support many modern optimization strategies. Fluere will compile using the older GCC 4.2.1, but the resulting executable will be slow (for example, image deformation runs about four times faster on new versions of GCC). For this reason, I would recommend downloading the latest GCC binaries

and following the installation instructions at http://hpc.sourceforge.net. If you have questions about this procedure please contact me and I can walk you through the process.

Note: If you successfully compile the code on the Windows machine, but the program crashes each time you try to launch it, this may be due to `pthreadGC2.dll` missing on your computer. This is a library needed for the multiprocessor capabilities of Fluere. If you need a copy of it, it is available in the binary distribution. Copy this into the same directory as the executable and the program should run.

2 Getting Started

Upon launching the program, you will be presented with a window containing the menu bar and two docked windows. The side dock contains the listing of image files to be processed, and the lower dock provides text output and information concerning the processing.

2.1 Loading and Viewing Files

Images can be loaded by choosing the menu item `File > Open Image Files`. A dialog will appear allowing you to select one or more image files from a directory. If you'd like to open multiple files, Shift-click or Control-Click to select a range of files. The following image formats are currently supported:

TIFF The tagged image file format. Fluere supports both 8-bit and 16-bit grayscale TIFF files.

BMP The bitmap image format.

Once you've selected the files, each file name will appear in the file list window. You can either double-click on an file name to view the image, or use the options in the `Image` menu. This can make it easy to visualize the particle motion between images.

Fluere supports paired and combined input images. If it appears that only half of an image is being shown, or that two images are being shown at the same time, you may need to modify these settings as described in section 3.1. To alternate between the first and second frame of a combined image, use the options in the `Image` menu.

2.2 Performing the Correlation

Fluere uses conservative default settings which should give satisfactory results for properly acquired PIV data. Of course, these settings can be modified as described in section 3.2. Begin processing by selecting a pair of files (for split mode) or a single file (for combined mode) in the file listing and clicking `Process > Process Selected Files`. Fluere will test to ensure that the images are the correct size and that the correlation settings are valid, then proceed to the interrogation. During this time you will see a series of messages in the console window which indicate the progress.

When the software is finished processing an image pair, the vector field will be added overlaying the image. The display properties, such as vector color and length, can be modified as described in section 3.4. Vectors that were flagged as invalid by the outlier detection algorithm, and subsequently interpolated, will be colored white. Note that the vectors may no longer look invalid since they have been interpolated using surrounding data, but the highlighting helps inform the user of the quality of the data set. To show the raw data before interpolation has been performed, click `Plot > Draw Raw Vectors`. An example of a correlation is shown in figure 1.

Performing batch processing on a large group of files is simple, and can be done by selecting a group of files in the file list. Note that for split mode you must select an even number of files. Fluere will warn you in case this criteria is not satisfied. In batch mode, each image will be processed and displayed. If processing must be stopped at any point, it can be terminated by selecting `Process > Stop Processing`.

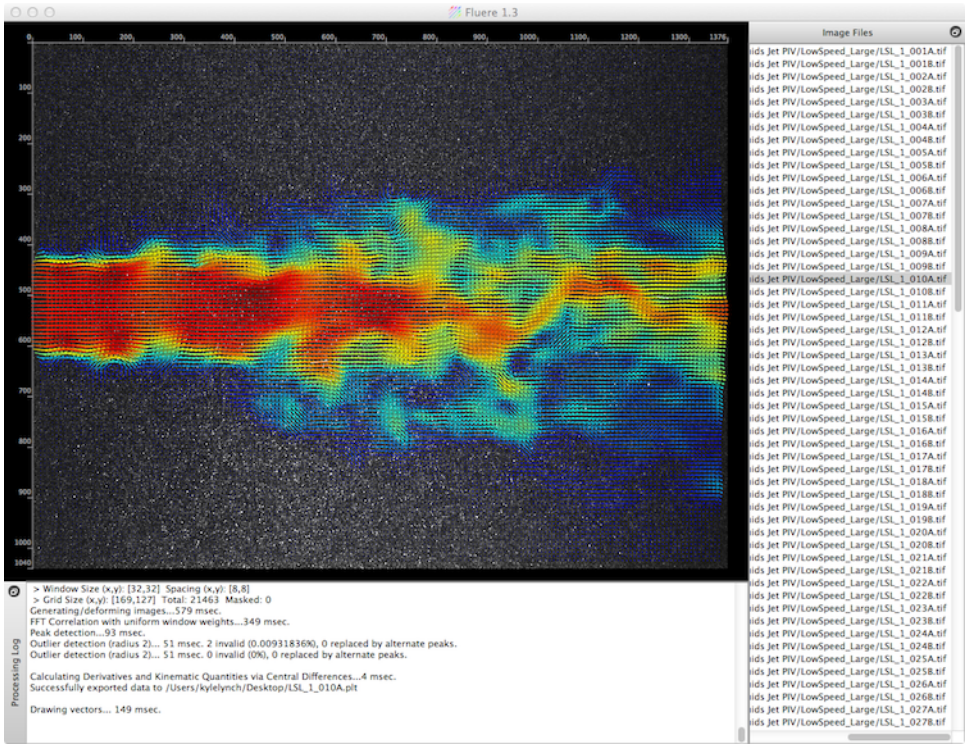


Figure 1: Screenshot of the processed vector field of a turbulent jet.

Fluere offers two other modes of processing besides the standard processing settings. The first, ensemble correlation, sums the correlation planes from a large set of images to increase the robustness of the displacement estimate in steady flows. It is emphasized that this technique is only valid for steady flows.

The other technique is multiframe correlation, where a sliding ensemble correlation is used over a small kernel of images to make the displacement estimate more robust. Also, the result of the correlation is used as the predictor for the next correlation in the sequence, eliminating the need to run the first few passes. Note that this technique requires a time-resolved image sequence and only split images can be used.

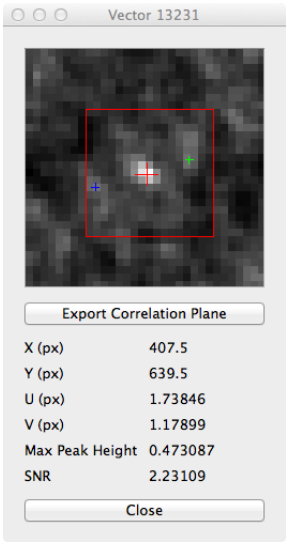


Figure 2: Screenshot of a vector information dialog.

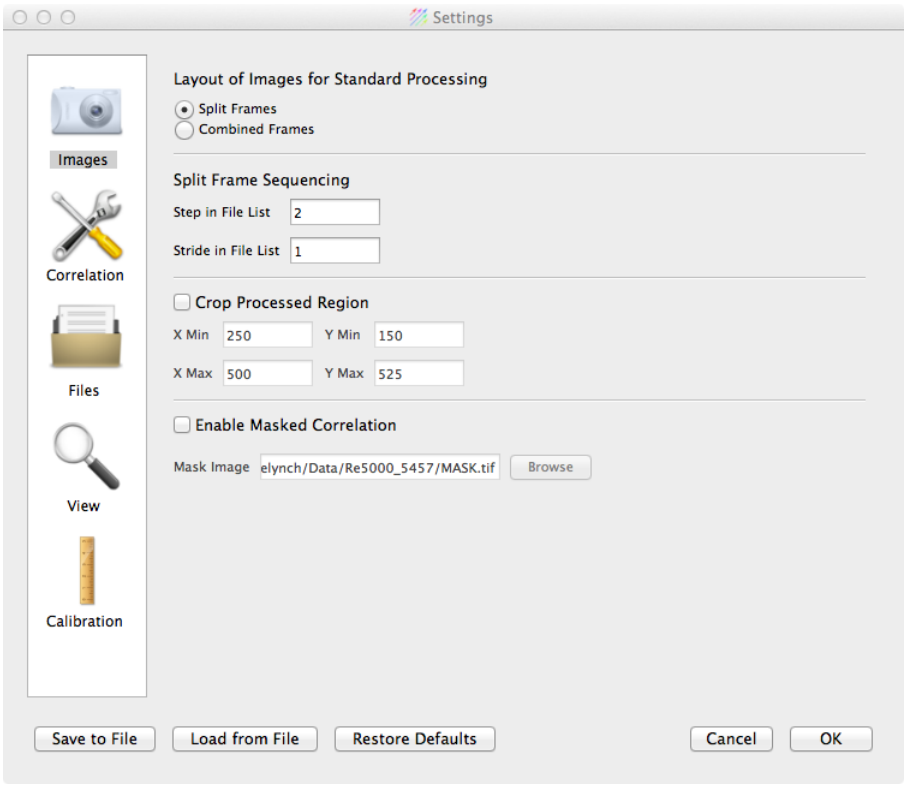
2.3 Examining the Correlation

Fluere was designed to provide a wealth of information regarding the correlation to the user. Internally, the program saves all correlation planes to provide a unique capability for analysis after the correlation is complete. When the mouse is placed over a vector, the vector will become bold. Clicking on the vector will raise a dialog giving vector information and the correlation plane. This is useful for both educational purposes and for evaluating the robustness of the correlation. Overlaying the correlation plane is a large rectangle which indicates the extent of the peak search radius, and up to three colored crosses indicating the first (red), second (blue), and third (green) peaks. The largest cross indicates the peak used for the velocity estimate. For example, if the blue cross is largest, it indicates the second peak was identified as the true peak in the outlier detection algorithm. If no peaks are large, it indicates that the vector was interpolated from surrounding

data. An example of this dialog is shown in figure 2. Note the correlation plane can be exported. This is useful for performing post processing, in which the exported text file can be easily loaded into Matlab for analysis using the `load -ascii` command.

3 Program Settings

3.1 Image Settings



Fluere supports two layouts for the input images.. The two types of images supported are:

Split Frames A separate image file for image A and image B. The files must be the same dimensions.

Combined Frames Image A and image B located within the same image file, tiled one on top of the other. Internally, image A and image B are extracted by splitting the image file.

For the split frame mode, Fluere offers options on how to step through the file list. *Step in the file list* will indicate to the software the number of files between adjacent image first frames. The *stride in the file list* indicates the separation in files between frame A and B. For example, a step of 2 and a stride of 1 would result in a processing sequence: $I_1 * I_2$, $1_3 * I_4$, etc. A step of 1 and a stride of 1 would result in a sequence: $I_1 * I_2$, $1_2 * I_3$, etc. As a final example, a step of 1 and a stride of 2 would result in a sequence: $I_1 * I_3$, $1_2 * I_4$.

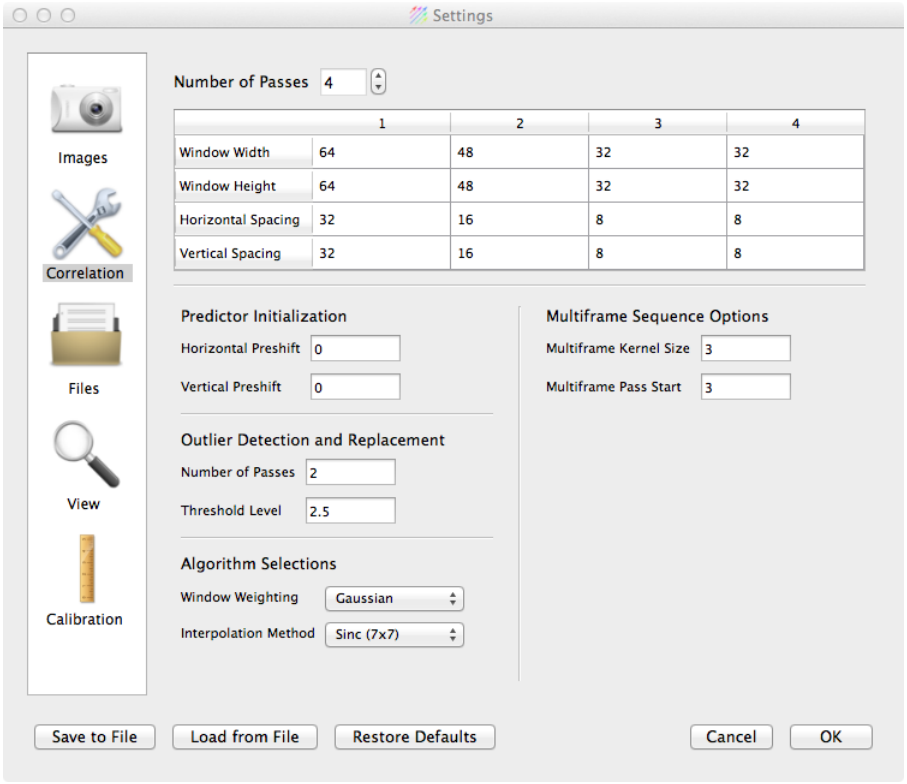
Cropping the processed region can be used to process only a small region of interest in the images without having to crop the images themselves. The values input into this section specify the top-left and bottom-right points of the rectangle defining the cropped region.

Fluere also supports image masking, which allows vectors which overlay solid surfaces or interfaces to be ignored. This is done by specifying a mask image where the gray levels of the areas to be masked are set to 0. Internally, Fluere takes the centroid position of the interrogation window

and compares it to the corresponding location in the mask image to determine whether the vector should be masked. For both paired images and combined images, only one mask is needed which is equal to the size of an individual image (i.e., for a combined image, a mask image only needs to be generated for the upper half of the image and it will be equally applied to both images). As an additional enhancement, Fluere sets the image gray values of the masked regions equal to the mean gray level throughout the image to prevent masked areas from distorting the correlation. This drastically reduces bias errors on measurements near interfaces.

To create a masked image an external program can be used. I highly recommend the program ImageJ (http://imagej.nih.gov/ij/index.html). Simply use the polygon selections and fills to set parts of an image to zero, and the unmasked regions to a non-zero value. Save the image as a separate file and you're ready to go.

3.2 Correlation Settings



Fluere gives the user a high level of control over the correlation processing. The default settings will work well on most images; however, careful tuning of the parameters can yield better results depending on the image characteristics. These settings are described below.

Number of Passes Specifies the number of passes used in the multi-pass processing.

Window Width The number of pixels in the horizontal direction for each interrogation window.

Window Height The number of pixels in the vertical y-direction for each interrogation window.

Horizontal Spacing The number of pixels between windows in the horizontal direction.

Vertical Spacing The number of pixels between windows in the vertical direction.

Horizontal Preshift Initializes the predictor in the horizontal direction. This is useful for images with substantial bulk motion in a particular direction.

Vertical Preshift Initializes the predictor in the vertical direction.

Outlier Number of Passes Specifies the number of validation and replacement passes to be used. More passes return a smoother result.

Threshold Level Sets the threshold level for the normalized median test. See [5] for more details. Typically set to 2 or 2.5.

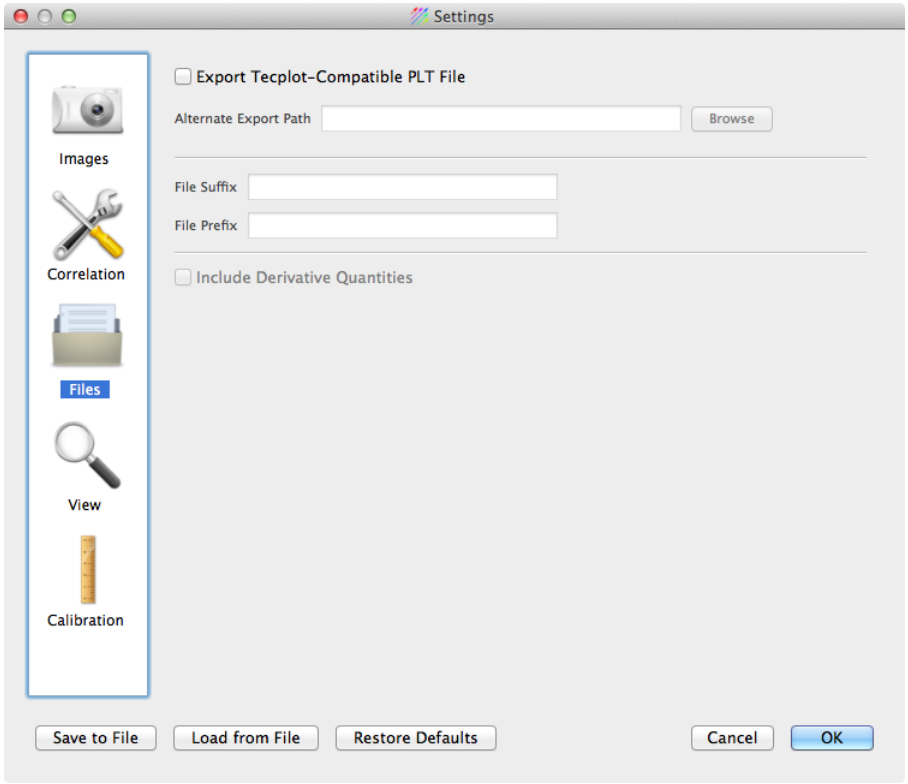
Window Weighting Selects the interrogation window weighting function to be used, either Uniform or Gaussian.

Interpolation Method Selects either a Bilinear, 7 x 7 Sinc, or 11 x 11 Sinc function for performing the image interpolation. Higher-order interpolators take more time to run, but may yield more accurate results. Refer to the validation portion of the manual.

Multiframe Kernel Size Number of correlations to average together for multiframe processing.

Multiframe Pass Start Specifies the pass that the multiframe processing starts with once the initial predictor is generated.

3.3 File Settings



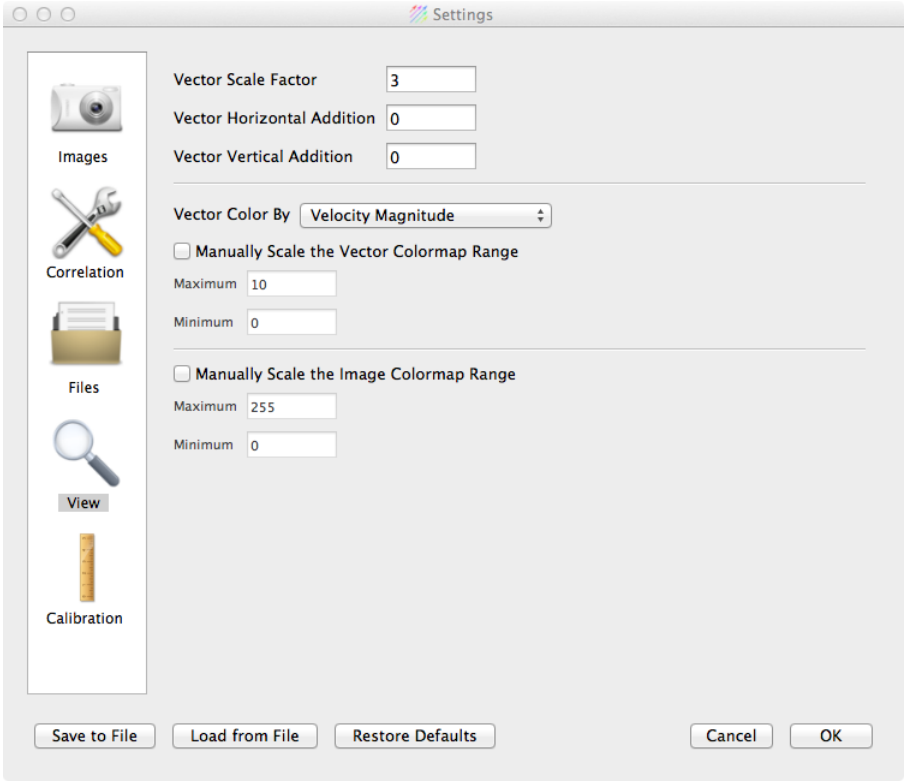
By default, Fluere does not export any data in order to prevent cluttering up a user's hard drive. To save the data, check *Export to Tecplot-Compatible Data File*. This will export the data file using the same file name as the image file, but with a different extension. For example, if an image file name is `C:/image_0001A.tif`, the output file will be saved as `C:/image_0001A.plt`. You can change the directory the files are exported to by specifying an alternate export path, and you can add a suffix and prefix to the file in the appropriate boxes.

The data file contains a three-line Tecplot header, which allows the file to be directly read in to Tecplot using the Tecplot Data Loader. The rest of the file contains the data in multiple columns,

with the variable names given in the header. To read the data into Matlab, the function `textread` can be used, or the function file `fluere_dat2mtx` can be downloaded from the Fluere website.

Fluere is also capable of saving derivative quantities in the data file. Fluere uses central differencing to calculate all first-order and second-order derivatives. Also reported is the angular rotation ω, and the strain ϵ_{xy}. Note, vorticity is just 2 times the angular rotation.

3.4 View Settings



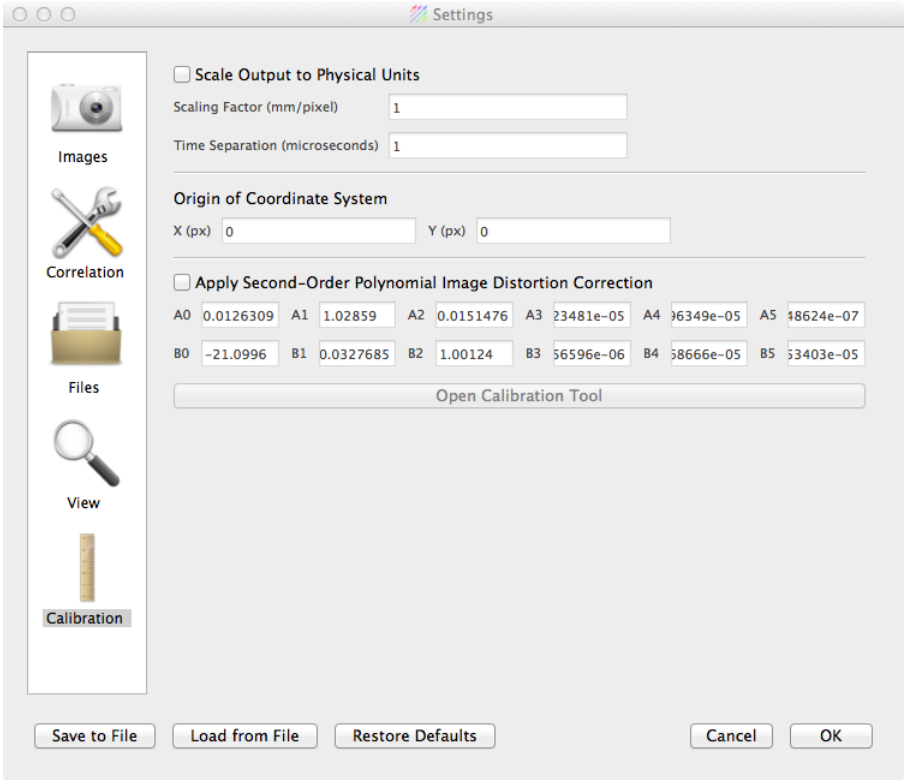
There are multiple settings which can be used to modify the vector display. The vector scaling factor determines the length of the vectors themselves. The horizontal and vertical additions allow for individual components to be modified for display purposes only (this does not affect the output data whatsoever). For example, when visualizing a turbulent boundary layer, much of the structure can not be easily seen due to the large freestream velocity. By subtracting around 80% of the freestream displacement, clear vortical structures are able to be seen.

The vector color scaling can be adjusted to represent velocity magnitude, value of both velocity components, and correlation signal-to-noise ratio. In each of these cases, the color scale minimum and maximum can be set manually, or *Manually Scale the Vector Colormap Range* will automatically scale the colormap to the maximum and minimum values of the variable.

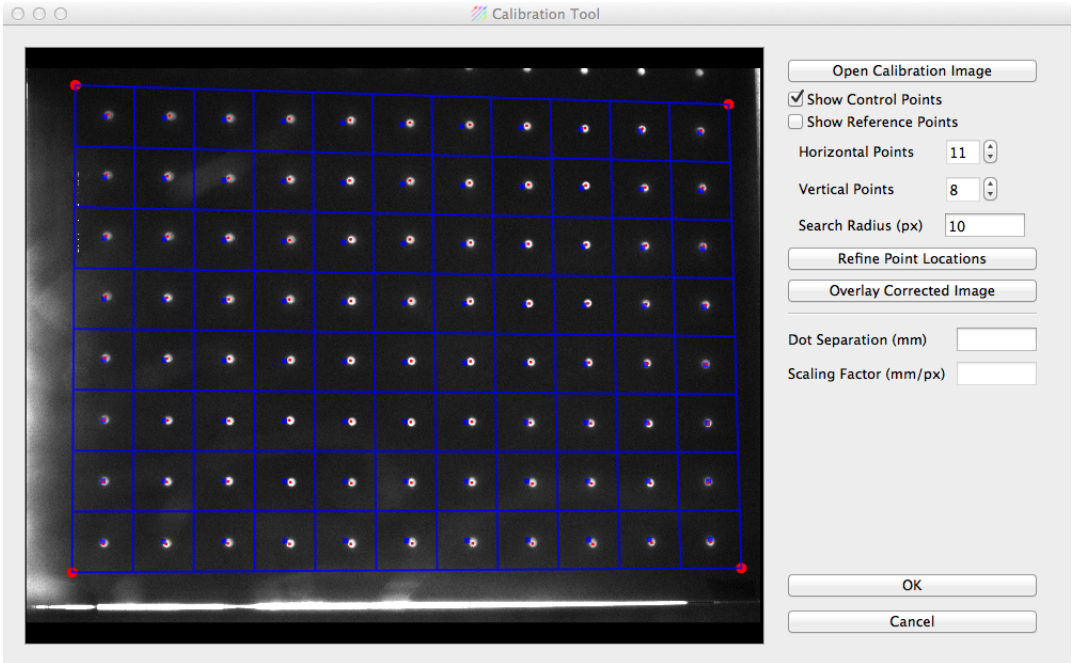
Likewise, the image colormap is by default automatically scaled. However, it can be manually set using the *Manually Scale the Image Colormap Range* option.

3.5 Calibration Settings

Fluere can automatically convert pixel displacements to physical velocities by inputting the correct physical parameters. The time separation between images must be stated in microseconds and the conversion factor must be specified millimeters per pixel. The latter can be determined through visual inspection of the images or through the calibration tool as described below. Note that only the output exported to a file will be scaled.



Another feature of Fluere is the ability to move the origin of the axes to an arbitrary location, which may aid when axes need to be aligned to a specific feature in the image. The origin of the coordinate system will be moved to the location specified here in pixels.



To correct for image misalignment or image distortions such as pincushion or barrel distortion, Fluere has implemented a calibration system based on a second-order image mapping technique.

If your images need to use this more advanced calibration step, check *Apply Second-Order Image Mapping with Calibration Tool*, and open the calibration tool.

Once the calibration tool has been opened, you will first want to load in a calibration image which consists of a dot target. The dots must be high intensity regions. If your dot image consists of dark dots, use an external program to invert the image.

The first step is to manually count the dots along both the horizontal and vertical directions and input this information into the horizontal and vertical points inputs. This resizes the number of cells in the grid to match your images. Then, grab each of the four red control points and drag them such that the blue dots at the center of each cell match approximately the dot locations. To see the grid which the images will be transformed to fit to, click the *Show Reference Points* check box.

Once the grid is positioned, click *Refine Point Locations* to determine the dot locations to greater accuracy using a centroid search algorithm. If some of the dots are far from the blue initial dots, you may need to increase the search radius. To see the final effect of the mapping, click "Overlay Corrected Image" to show the new, undistorted image which represents the same correction to be applied to all images.

4 Algorithms

The processing engine of Fluere is based on the well-established iterative image deformation algorithm described by Scarano and Riethmuller [4]. This algorithm greatly extends the dynamic range and accuracy of velocity measurements by using an image deformation scheme that accounts for the displacement and displacement gradients within interrogation windows. The method also drastically reduces the effects of peak-locking compared to traditional single-pass or discrete window offset algorithms. A more detailed description of the algorithm components is provided herein.

4.1 FFT-Based Correlation

The discrete representation of the cross-correlation function is given in equation 1, where CC is the correlation plane, I_A is the interrogation window from the first image, I_B is the interrogation window from the second image, and p and q are the shift of the windows relative to each other. Equation 1 can be evaluated using a direct approach; however, this is computationally intensive and therefore the approach most widely used utilizes the Fast Fourier Transform (FFT). A proof of the equivalence of the direct and FFT correlation can be found in Adrian and Westerweel [1].

$$CC[p,q] = \frac{1}{M \times N} \sum_{i=1}^{I} \sum_{j=1}^{J} I_A[i,j] I_B[i+p, j+q] \tag{1}$$

The calculation of the cross-correlation in Fluere consists of the following steps for each interrogation window:

1. Image windows I_A and I_B are extracted. For Gaussian weighting, this is done by extracting a window *twice* the size of the desired window in each direction (i.e., for a desired 32 x 32 window, a 64 x 64 region is extracted).

2. The mean value of each image window is then calculated and then subtracted.

3. A window weighting function is applied. For uniform weighting, this is a top-hat function equal to the size of the desired window. For Gaussian weighting, the values of the Gaussian function are used to weight the windows, such that the values at the edge of the desired window are equal to e^{-1}.

4. The real-to-complex FFT of both interrogation windows is performed to yield I_A^* and I_B^*.

5. I_B^* is multiplied by the complex conjugate of I_A^* to yield CC^*.

6. The complex-to-real inverse FFT is performed on CC^* to yield CC.
7. CC is reshuffled to place the DC-component in the center of the image. This is accomplished by swapping quadrants 1 and 3, and 2 and 4, of the correlation plane (equivalent to the `fftshift` command in Matlab).
8. The correlation plane is divided by the auto-correlation of the weighting function. This corrects for window bias errors and properly scales the result.

The above steps may seem complex, but can be very efficiently computed compared to a direct correlation approach. An example of the inputs to this process and the resulting correlation plane is shown in figure 3.

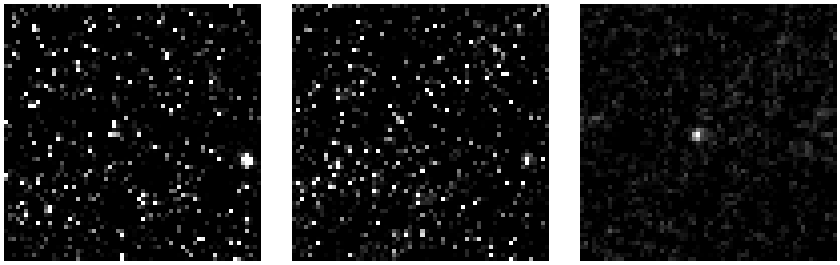


Figure 3: I_A (left), I_B (center), and CC (right). Example using experimentally acquired image data and window sizes of 64 x 64 pixels.

4.2 Peak-Finding Algorithm

Fluere uses a multiple peak finding algorithm as outlined in Raffel et al. [2] to extract the maximum amount of information from each correlation plane. The first step scales the correlation plane such that each value lies above zero. Next, the correlation plane is scanned within a restricted search radius to find pixels that comprise a peak of their surrounding 4-connected neighborhood. This search radius is set to one-quarter of the window size. The use of this restricted search radius prevents inappropriate peaks from being identified and reduces the number of obvious outliers generated. Note that this also requires the user to select the initial window size and/or preshift in accordance with the one-quarter rule such that the peak will occur within the search radius. Once the collection of peaks is found, they are sorted based on the intensity value of the peak. The highest three peaks are stored, and the remaining peaks are discarded.

Next, Fluere uses a three-point Gaussian estimator to refine the displacement estimate to sub-pixel accuracy. It has been proven that the use of a Gaussian fit compared to centroid or parabolic fitting results in the lowest magnitude of bias errors and random errors [3]. Two independent fits are performed in both the x and y directions. Equation 2 is used for the calculation, where R_0 is the peak value, $R_{+1,-1}$ are the neighboring values, and d_0 is the integer location of the peak value within the search radius of the correlation plane.

$$d = d_0 + \frac{\ln R_{-1} - \ln R_{+1}}{2\ln R_{-1} - 4\ln R_0 + 2\ln R_{+1}} \tag{2}$$

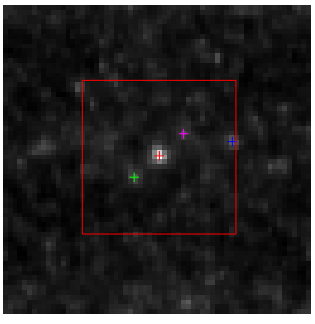


Figure 4: Correlation plane example.

An example of the restricted search radius and multiple peak-finding is shown in figure 4. The search radius is outlined, which indicates the central region of the correlation plane is considered valid for evaluating displacements. The regions outside of this radius are not searched. Also, the sub-pixel estimate of the displacement is shown in red. Other peaks are identified in various colors.

The use of this equation poses two requirements on peak and neighboring values. First, the data must all be positive values, that is, $R_{-1,0,+1} > 0$, since the value of the natural logarithm is undefined for zero or negative inputs. For this reason, The second requirement is that the value at R_0 comprises a peak, that is, $R_0 > R_{-1,+1}$. These requirements are always satisfied due to the nature of the peak searching algorithm described above.

After the peaks are sorted by peak height the velocity field is initialized using the peak with the maximum height. In the validation steps described in the next section, the additional peak data can be used as a replacement for a spurious maximum peak.

4.3 Outlier Detection and Replacement

Iterative PIV processing requires the elimination of spurious or invalid vectors after each pass to prevent subsequent passes from being initialized with erroneous information. Fluere implements a configurable multi-pass vector validation and replacement technique. To identify invalid vectors, the normalized median test is used. This is an extension of the median test which normalizes the median residual by an estimate of the local variation of the velocity, and is described in detail by Westerweel and Scarano [5]. Fluere implements the test using a user-selectable threshold level (default 2.5), a minimum fluctuation level of 0.1, and a 5 x 5 kernel.

The test is performed at each interrogation location by looking at the representative velocity vectors formed by the three highest peaks as previously identified. If the maximum peak corresponds to an invalid vector, the second and third peaks are evaluated in order to determine if they pass the test. If a peak is found that passes the test, the velocity vector is updated to correspond to that particular peak. This procedure can recover a substantial number of the invalid vectors and helps to extract the maximum data yield from PIV images. The remaining vectors that do not contain valid peaks are replaced by performing an average of the four nearest valid vectors.

This procedure can be repeated multiple times (default 2 times) to smooth the data to a desired level.

4.4 Image Interpolation Scheme

For image deformation, the predictor is defined for each pixel of the image through a bilinear interpolation of the displacement at each pixel location (this is known as the dense predictor field). The values of the dense predictor are non-integer values and thus require the image intensity to be evaluated at non-integer locations. The choice of interpolation function has been studied by Astarita and Cardone [7], where it has been determined that a sinc function interpolation (also known as a Whittaker interpolation) typically results in the smallest amount of error in the correlation, due to the fact that the sinc function is considered an ideal interpolation scheme in the context of frequency response. Thus, Fluere uses a user-selectable 7 x 7 (default) or 11 x 11 sinc filter to perform the interpolation. Additionally, Fluere offers a bilinear image interpolation option for much faster results, but with decreased accuracy. A comparison of these schemes is given in the validation section of the manual.

4.5 Predictor Filtering

Critical to iterative PIV evaluation is the suppression of oscillations in the result. A widely used technique is to apply a filter on the predictor to stabilize the algorithm. In Fluere, a second-order spatial regression is used. This has been shown to yield a combination of high spatial resolution and low RMS errors by Schrijer and Scarano [8]. The spatial regression is performed over a 5 x 5 kernel.

4.6 Image Calibration

In perfect imaging conditions, geometric optics can be used to determine the simple, linear correspondence between the image plane and the object plane. However, if a camera is viewing the

object plane from an angle or if optical nonlinearities are present (e.g., pincushion and/or barrel distortion), a more sophisticated mapping is required to prevent these distortions from affecting the measured displacements.

A general correction approach can be achieved using a second-order image mapping, which defines the transfer function between object coordinates (X, Y) to image coordinates (x, y) by the following equations:

$$X = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 \quad (3)$$

$$Y = b_0 + b_1 x + b_2 y + b_3 x^2 + b_4 xy + b_5 y^2 \quad (4)$$

If at least six points are known in image space and object space, the a and b coefficients can be determined by solving a linear system of equations.

Once these sets of coefficients are formed, the shift for each pixel can be calculated to align the image coordinates to the proper object coordinates. This shift is implemented in the image deformation stage as an additional shift on top of the dense predictor. In this manner, excessive smoothing by multiple interpolations is avoided while introducing very little computational overhead in the processing.

5 Validation

To ensure the algorithms are working consistent with reported performance in the literature, a series of validations have been performed to assess the accuracy and spatial resolution of Fluere. To generate the synthetic particle images in this validation, a technique based on the EUROPIV Synthetic Image Generator (SIG) was used [6], which integrates a Gaussian intensity distribution over a synthetic sensor. Images of size 512 x 512 pixels were used. The particle density is set to 0.1 particles per pixel, and the Gaussian standard deviations of the particle images were set to a uniform value of 2.0 pixels. The particle intensity and laser sheet profile were taken to be uniform, and the fill factor of the CCD was set to 100 percent.

5.1 Synthetic Uniform Displacements

A series of images were generated to simulate uniform displacements from 0 to 3.5 pixels in increments of 0.1 pixels. The purpose of this testing is to evaluate the exact error for high-quality data and ensure the image interpolation algorithms meet theoretical performance. The images were processed using Fluere and the resulting data files were analyzed in Matlab. Figure 5 presents the mean error of all the image interpolation methods. This is defined as the reported displacement minus the actual imposed displacement. Figure 6 provides the total error, defined as the RMS of the reported displacement minus the actual imposed displacement. To compare this data to published literature, refer to the review by Astarita and Cardone [7]. Comparisons can also be made with the paper of Scarano and Riethmuller [4]. Note that the conventional correlation is performed using a single pass of 32 px^2.

In practice, these error estimates are overly optimistic since the data used for this validation is synthetically generated and free from errors. Do not use these values as an error estimate for real PIV data, only as an indication of general trends in the error and as a statement that the algorithms of Fluere work as designed.

5.2 Synthetic Sinusoidal Displacements

The above analysis of uniform displacements is useful for assessing the accuracy of the image interpolation techniques and the sub-pixel displacement estimation algorithm. However, most PIV investigations are concerned with flows containing a wide range of spatial scales, some of which may be on the order of the interrogation window size. Thus, it is important to define the modulation transfer function of the algorithm to properly describe the attenuation that small scales will experience.

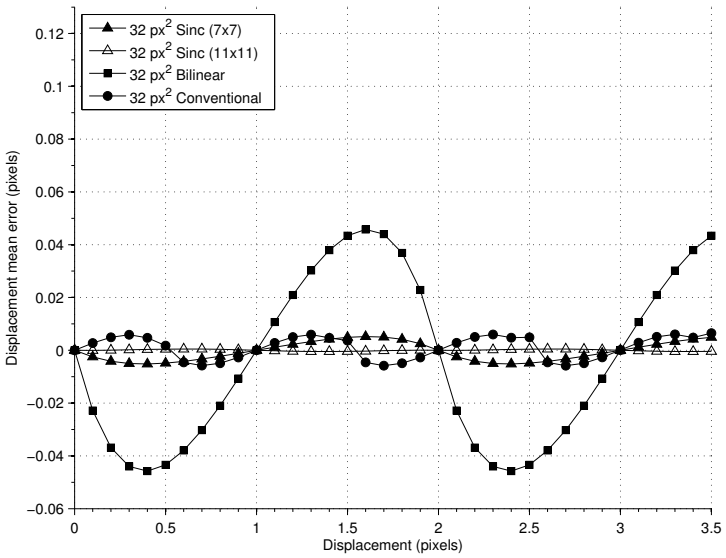


Figure 5: Displacement mean errors for uniform displacements from 0 to 3.5 pixels, in increments of 0.1 pixels.

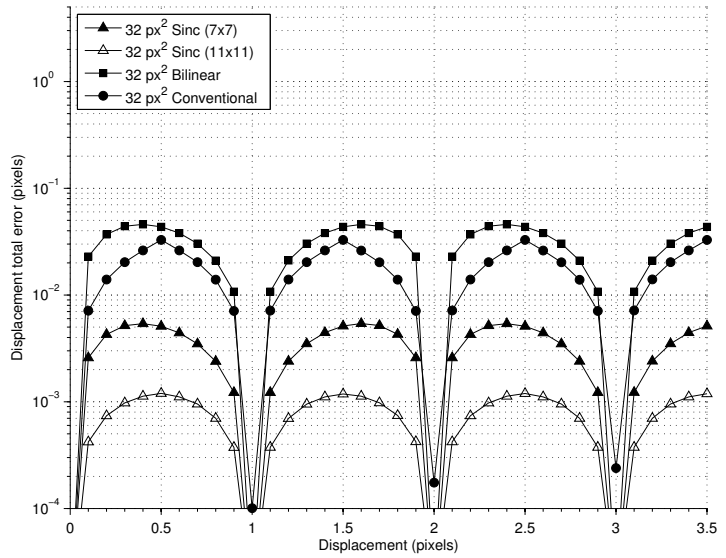


Figure 6: Displacement RMS errors for uniform displacements from 0 to 3.5 pixels, in increments of 0.1 pixels.

The use of interrogation windows in the PIV processing is roughly analogous to the process of using a fixed-width moving-average filter on a continuous signal. As outlined previously, depending on the frequencies present in the signal, the filter will attenuate the signal by a certain amount. To analyze this attenuation, a set of synthetic images were generated using a displacement field given by equation 5, where u is the imposed displacement, A is the peak amplitude (set to 1 pixel), and Λ is the wavelength.

$$u(y) = A\sin(2\pi y/\Lambda) \tag{5}$$

The images were processed using Fluere with window sizes of 32 x 32 pixels, window spacing

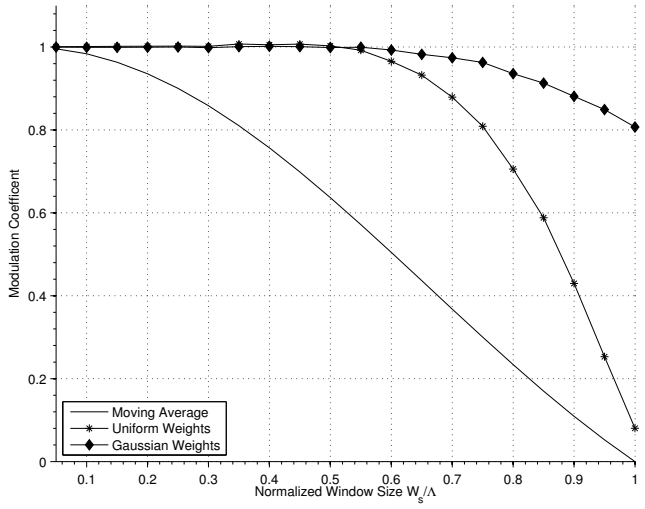


Figure 7: Modulation coefficient for sinusoidal displacements of varying wavelength.

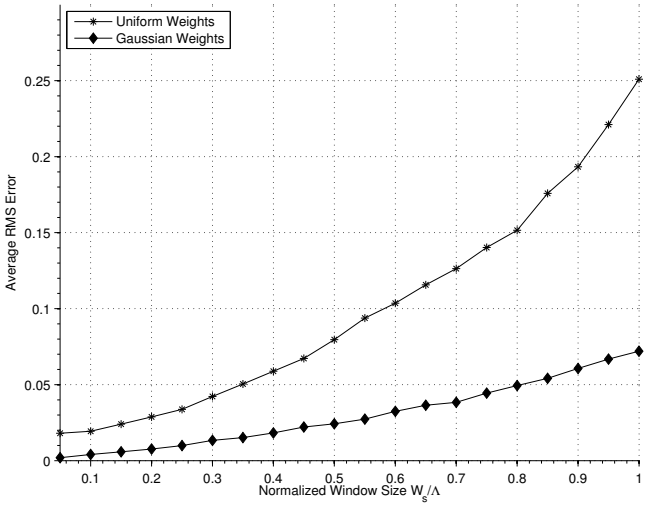


Figure 8: Average RMS error for sinusoidal displacements of varying wavelength.

of 3 x 3 pixels, and the 11 x 11 Sinc image interpolator. The resulting analysis is mirrored on the work by Schrijer et al. [8]. The displacements are evaluated in Matlab, where the columns were averaged to reduce noise. To further reduce noise, a numerical integration is performed which evaluates the area under the measured displacement curves with the area under a calculated sine function. Additionally, the wavelengths are normalized in terms of the window size for simpler interpretation. The modulation coefficient is shown in figure 7. A sinc function is also presented to represent the frequency response of a moving-average filter. Note that Gaussian weighting helps to increase the range of frequencies than can be reliably captured. Thus, for flows with spatial scales on the order of the window size, it is highly recommended to use the Gaussian window weighting. Also presented is the average RMS error generated by the algorithm, presented in figure 8. Here it is also clear to see the effects of the Gaussian window weighting in reducing the errors as the frequencies present in the displacement field increase.

5.3 Experimental Data

For the sake of completeness, a final validation is performed on experimental images to test the algorithm on real data. The sample data case was chosen to be a publicly available data set, case A from the 2003 PIV Challenge. This is a set of 100 image pairs of a submerged turbulent jet which are provided in a preconditioned form with contrast normalization applied. An example of a single interrogation is shown in figure 9, with one out of two vectors shown for clarity. For the tests conducted herein, the default settings of Fluere are used; however, the 11 x 11 sinc interpolation is chosen.

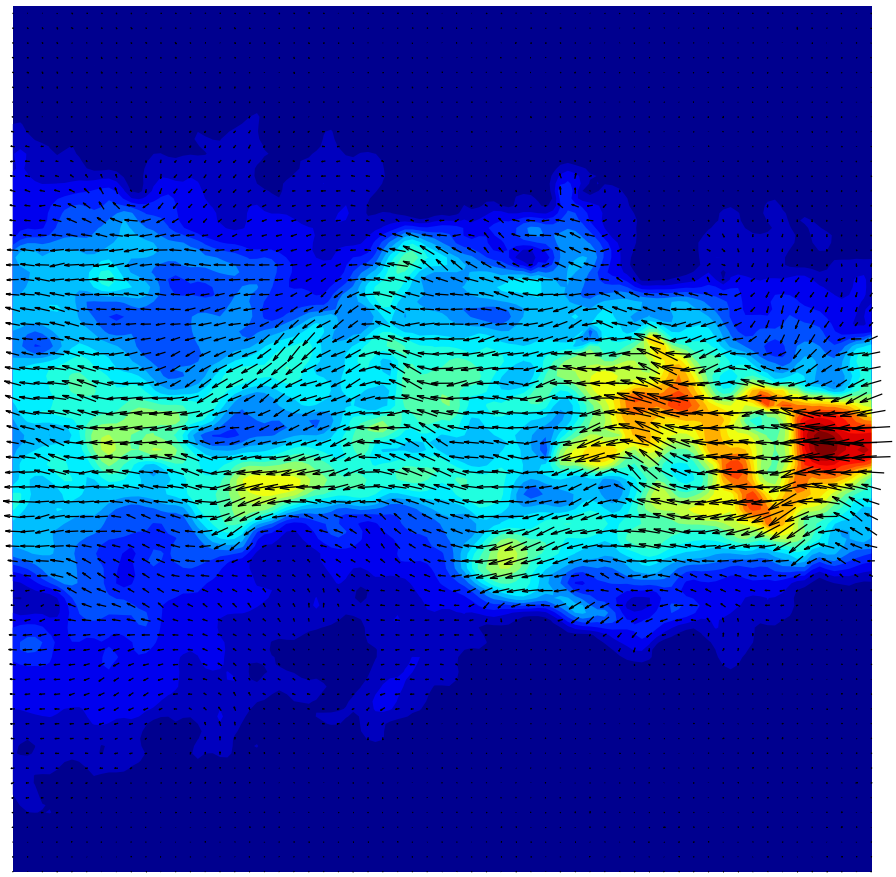


Figure 9: Example of a single instantaneous turbulent jet velocity field processed using Fluere. One out of every two vectors is plotted for clarity. Contour levels indicate velocity magnitude. Data is the image pair A001 of the PIV Challenge turbulent jet data.

Figures 10 through 14 illustrate the statistics produced by Fluere over the set of 100 measurements. Note that 100 measurements is statistically small, so the convergence of some quantities may appear suspect. However, the proper general trends are observed, such as the behavior of the Reynolds stress.

As a final evaluation, the effect of peak locking is determined by plotting histograms of the displacement. Figures 15 and 16 show the histograms for the u- and v-components, respectively. Note that a small amount of peak locking is seen in the u-component, which could be attributed to the image data itself being improperly sampled. A check of peak locking should be a part of every PIV investigation to ensure proper a proper imaging configuration is being used.

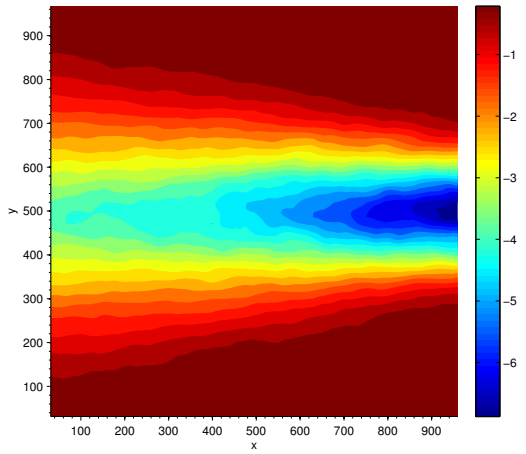


Figure 10: Average U Velocity

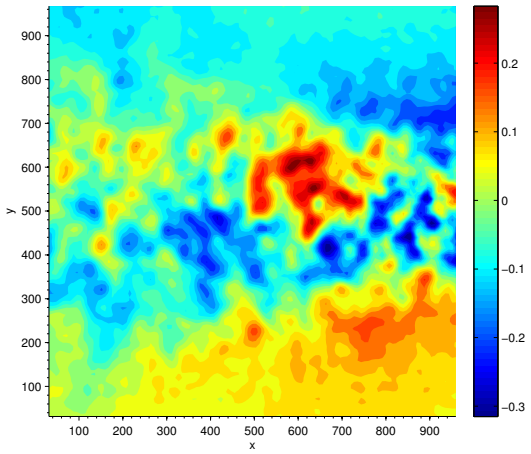


Figure 11: Average V Velocity

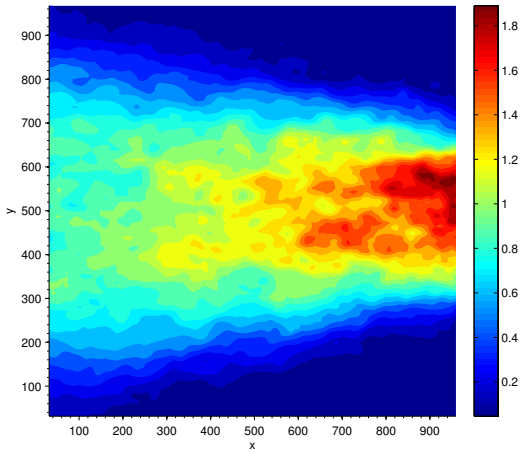


Figure 12: RMS of U Velocity Fluctuations.

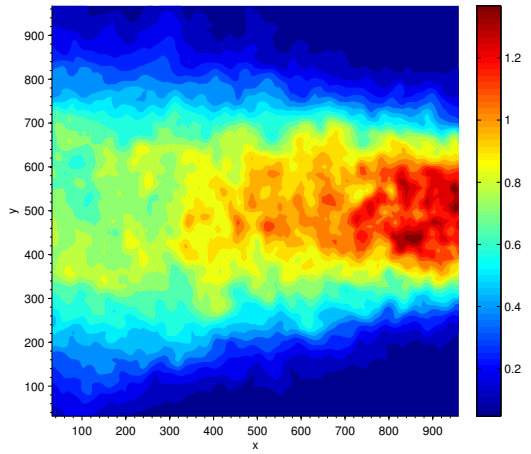


Figure 13: RMS of V Velocity Fluctuations.

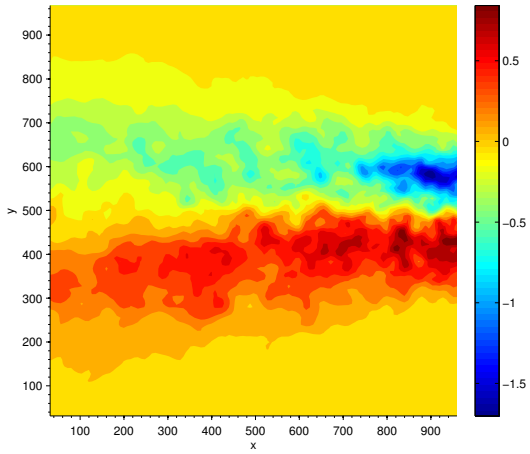


Figure 14: Average Reynolds Stress.

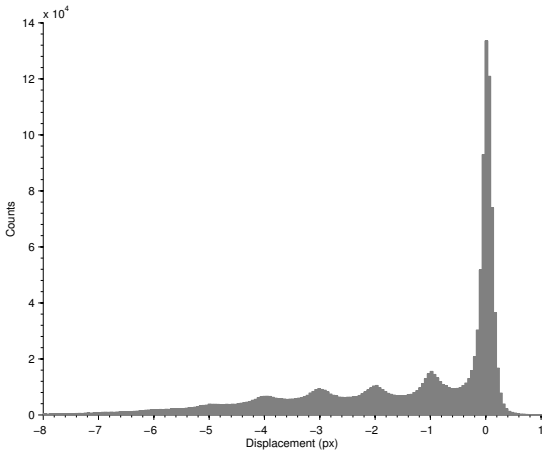


Figure 15: Histogram of displacements in the horizontal direction.

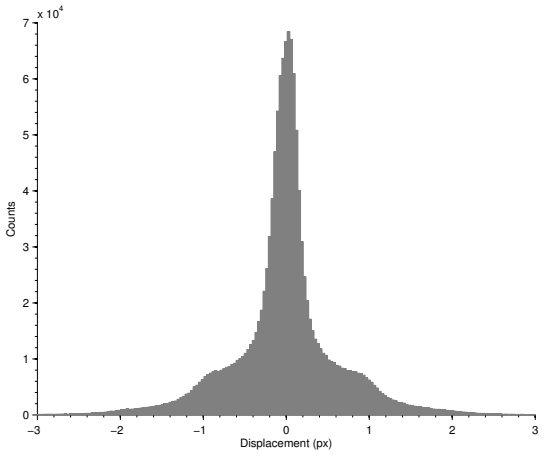


Figure 16: Histogram of displacements in the vertical direction.

References

[1] Adrian, R. and Westerweel, J., *Particle Image Velocimetry*, 2011.

[2] Raffel, M., Willert, C., Wereley, S., and Kompenhans, J., *Particle Image Velocimetry: A Practical Guide*, Second Edition, 2007.

[3] Westerweel, J., "Fundamentals of digital particle image velocimetry," *Measurement Science and Technology*, 1997.

[4] Scarano, F. and Riethmuller, M., "Advances in iterative multigrid PIV image processing," *Experiments in Fluids*, 2000.

[5] Westerweel, J. and Scarano, F., "Universal outlier detection for PIV data," *Experiments in Fluids*, 2005.

[6] Lecordier, B. and Westerweel, J., "The EUROPIV Synthetic Image Generator (S.I.G.)," *Proceedings of the EUROPIV 2 Workshop on Particle Image Velocimetry*, 2004.

[7] Astarita, T. and Cardone, G., "Analysis of interpolation schemes for image deformation methods in PIV," *Experiments in Fluids*, 2005.

[8] Schrijer, F. and Scarano, F., "Effect of predictor-corrector filtering on the stability and spatial resolution of iterative PIV interrogation," *Experiments in Fluids*, 2008.