Exp Fluids (2011) 50:1169–1182
DOI 10.1007/s00348-011-1054-x

RESEARCH ARTICLE

Fast and accurate PIV computation using highly parallel iterative correlation maximization

F. Champagnat · A. Plyer · G. Le Besnerais · B. Leclaire · S. Davoust · Y. Le Sant

Received: 31 December 2009 / Revised: 26 January 2011 / Accepted: 1 February 2011 / Published online: 17 March 2011

Abstract Our contribution deals with fast computation of dense two-component (2C) PIV vector fields using Graphics Processing Units (GPUs). We show that iterative gradient-based cross-correlation optimization is an accurate and efficient alternative to multi-pass processing with FFT-based cross-correlation. Density is meant here from the sampling point of view (we obtain one vector per pixel), since the presented algorithm, FOLKI, naturally performs fast correlation optimization over interrogation windows with maximal overlap. The processing of 5 image pairs (1,376 × 1,040 each) is achieved in less than a second on a NVIDIA Tesla C1060 GPU. Various tests on synthetic and experimental images, including a dataset of the 2nd PIV challenge, show that the accuracy of FOLKI is found comparable to that of state-of-the-art FFT-based commercial softwares, while being 50 times faster.

1 Introduction

Particle Image Velocimetry (PIV) has become an essential tool for flow diagnosis and is therefore widely used in industrial as well as academic situations. Its current limitation is however, the time necessary to compute the vector fields from the images, which often imposes specific constraints in the schedule of test campaigns. In that respect, the important development of high-speed PIV systems over the last decade appears even more challenging. We propose a solution to shorten dramatically this processing time, based on an algorithm that computes dense 2C vector fields using Graphics Processing Units (GPUs).

GPU has already been compared to other architectures for PIV processing in previous works (Schiwietz and Westermann 2004; Venugopal et al. 2009). These studies concentrated on cross-correlation using FFT, but the speed-up factor for FFT using GPU versus CPU architecture does not exceed three. In this context, real-time computation therefore requires large PC clusters with a GPU at each node (Venugopal et al. 2009). Former real-time realizations also involve parallelization on Field-Programmable Gate Arrays (FPGA) (Iriarte Munoz et al. 2009; Yu et al. 2006). Although efficient and convenient for embedded systems, this solution is far more expensive than GPU to implement, both in terms of hardware cost and software development effort. Interestingly, these architectures get rid of FFT in favor of direct correlation, which is better suited to FPGA architectures. In contrast to these works, the approach proposed hereafter relies on a technique for cross-correlation maximization that departs from the classical FFT method or from direct correlation. Its structure is ideally matched to massively parallel architectures and therefore allows a 50 times speed-up using a single GPU.

The algorithm FOLKI (French acronym for Iterative Lucas–Kanade Optical Flow, Le Besnerais and Champagnat 2005) was originally designed in the context of computer vision for motion estimation in video sequences. But FOLKI proved also very robust and adaptive to many other kinds of images such as those obtained in photomechanics and PIV. It is based on the classical interrogation window

F. Champagnat (✉) · A. Plyer · G. Le Besnerais
Modeling and Information Processing Department,
French Aerospace Lab (ONERA), Chemin de la Hunière,
91761 Palaiseau Cedex, France
e-mail: fchamp@onera.fr

B. Leclaire · S. Davoust · Y. Le Sant
Fundamental and Experimental Aerodynamics Department,
French Aerospace Lab (ONERA), 8 rue des Vertugadins,
92190 Meudon, France

paradigm, but belongs to the family of Lucas–Kanade (LK) algorithms (see Baker and Matthews 2004, for a review). The basic LK method is already known in the PIV community but it is most often associated with Particle Tracking Velocimetry (Miozzi 2004; Stanislas et al. 2008), i.e., low-seeding densities and sparse estimation of displacements. In contrast, the improvement from the basic LK approach implemented in FOLKI naturally relies on the computation of dense fields, i.e., a displacement vector for each image pixel. This leads to a highly regular and parallel algorithm which is much more efficient than previous sparse LK techniques and furthermore specially suited to GPU architectures. Of course, the fact that one vector per pixel be obtained should not be confused with the spatial resolution of the method, which is tightly linked with the window size, as for any other window-based PIV technique.

The principle of FOLKI is the following: around each pixel, a fixed size interrogation window (IW) is defined, and a cross-correlation measure is defined as a Sum of Squared Differences (SSD) between the IW and a displaced window in the consecutive image. In contrast to mainstream PIV algorithms that perform extensive search over discrete pixel grid with a FFT correlation, this SSD is minimized using an iterative Gauss–Newton (GN) descent. On a general point of view, when initialized not too far (say 3 pixels) from the true displacement, it is known that the convergence of GN is fast, reaching a precision of the order of a tenth of a pixel in typically less than 5 iterations. As PIV images may often be characterized by larger displacements, a multiresolution scheme is used to avoid local minima. An image pyramid is built, starting from the acquired images, which correspond to the ground level. This is done by successively performing low-pass filtering and decimation, leading to successively smaller images. As each step also divides the displacements by two, this has to be done until the top-level images have displacements compatible with GN iterations initialized with a displacement equal to zero. This leads to first rough estimates, whose values are successively refined by descending the pyramid levels (whereby the spatial resolution is also refined). Such a coarse-to-fine multiresolution scheme has proven very efficient in optical flow methods in computer vision (Bergen et al. 1992) and is also used in PIV (see for instance Ruhnau et al. 2005). As will be shown in Sect. 2, an iterative image deformation technique (Lecordier and Trinite 2003; Stanislas et al. 2008) is implicitly embedded in the descent iteration.

Multiresolution, gradient descent and a dense velocity output are more often encountered in so-called "optical flow" methods (Corpetti et al. 2006; Ruhnau et al. 2005). However, FOLKI is a window-based method, with no spatial regularization such as a Horn & Schunk-like term (Corpetti et al. 2006). It should be compared to classical FFT correlators, which we will do in the assessment part of this paper.

The paper is organized as follows: Sects. 2 and 3 are devoted to a description of the basic FOLKI algorithm and to the principle of its GPU implementation. They consist in a more detailed version of the material presented in PIV'09 (Champagnat et al. 2009). Section 4 then describes specific improvements which were added to address situations typically encountered in PIV, the corresponding GPU implementation is then referred to as FOLKI-PIV. A detailed performance assessment follows, where FOLKI-PIV is characterized and benchmarked against a state-of-the-art commercial PIV software using FFT-based cross-correlation. First, synthetic images are specifically generated in order to determine its spatial resolution and its sensitivity to low-seeding densities and to noise. This is done in Sect. 5. Then, the comparative assessment is extended to experimental images, in Sect. 6. The level of peak-locking bias and sensitivity to actual measurement noise are explored by considering case A of the second PIV challenge (Stanislas et al. 2005), and results from a test campaign recorded at ONERA are introduced to show the advantages of dense sampling and illustrate how FOLKI-PIV deals with solid walls thanks to the use of masks. Finally, conclusive remarks and perspectives on future work are gathered in Sect. 7.

2 Basic FOLKI algorithm

2.1 Multiresolution setting and notations

The notations for the following derivations are illustrated in Fig. 1: observed image intensity at discrete positions $\boldsymbol{k} = [k, l]^t \in \mathcal{G} = \{0, \ldots, K-1\} \times \{0, \ldots, L-1\}$ and time indexes $t \in \{0, dt\}$ is denoted $I(\boldsymbol{k}, t)$. In the sequel, all summations $\sum_{\boldsymbol{k}}$ refer to summation on $\mathcal{G}$. We will sometimes use the notation $I(\cdot, t)$ for the function $\boldsymbol{k} \mapsto I(\boldsymbol{k}, t)$.

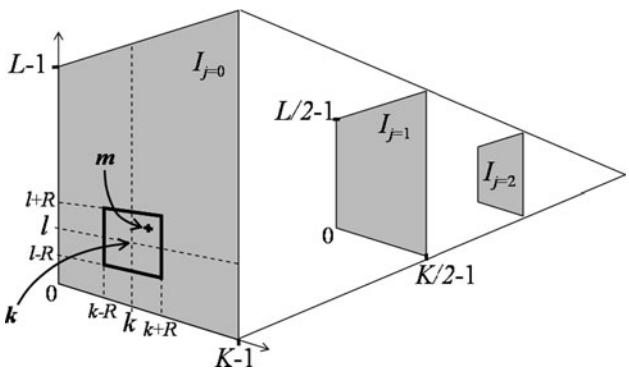

Fig. 1 Notations for image intensities within image pyramid and interrogation window

We use a multiresolution framework (Bergen et al. 1992): image intensity $I_j(\boldsymbol{x},t)$ at any real position $\boldsymbol{x}=[x,y]^t$, and any resolution level $j>0$ is computed by means of a Gaussian pyramid (Burt and Adelson 1983): starting from a level j, the image of level $j+1$ is obtained by applying a low-pass filter on the intensity $I_j(.,t)$ and then retaining one pixel out of a square of 2×2 pixels. Thus, image $j+1$ is four times smaller than image j, while displacements are divided by two. For each level, the spatial image gradient $\nabla I_j(\boldsymbol{x},t)$ is computed by a first-order centered difference scheme.

In this framework, displacements of the initially recorded image (level $j=0$) are thus divided by 2^j at level j. This allows to settle the question of the first estimate for the Gauss–Newton iterations: indeed, initialization at the highest level $J-1$ (where J is the total number of levels) can be done with zero displacement, as long as J is chosen so that displacements to find at level $J-1$ are sufficiently small in the whole image. In practice, for standard PIV images with an 8 pixel dynamic range, $J=3$ is enough for this process to work successfully, without being trapped in local minima.

2.2 A Lucas–Kanade algorithmic core

FOLKI relies on a Lucas–Kanade paradigm (Baker and Matthews 2004), which has been extended in Le Besnerais and Champagnat (2005) so as to provide a convergent iterative estimation of the dense displacement field $\boldsymbol{u}$.

In a majority of current PIV algorithms, the displacement of a given IW is found by first calculating the cross-correlation score of all possible displacements, then finding the maximum correlation peak, and finally refining its position by sub-pixel fit or interpolation. Usually, this process is repeated iteratively with decreasing window sizes, and at each step, the IWs are shifted using the previous estimation of displacement. The LK algorithm is also a window matching technique, but differs on both the objective criterion and on the way to obtain sub-pixel displacements. Cross-correlation maximization is in fact achieved by minimizing a Sum of Squared Differences (SSD), in which the displacement to be found appears directly as a real-valued (and not integer-valued) quantity. This is achieved thanks to a Gauss–Newton iterative descent. The SSD criterion around pixel $\boldsymbol{k}$ at level j writes

$$\sum_{\boldsymbol{m}} w(\boldsymbol{m}-\boldsymbol{k})\big(I_j(\boldsymbol{m},0)-I_j(\boldsymbol{m}-\boldsymbol{u}(\boldsymbol{k}),dt)\big)^2 \tag{1}$$

where w is a weight function whose support defines the interrogation window $\mathcal{W}(\boldsymbol{k})$:

$$\mathcal{W}(\boldsymbol{k})=\{\boldsymbol{m}\in\mathcal{G}\mid w(\boldsymbol{m}-\boldsymbol{k})>0\}. \tag{2}$$

The following derivations are valid for any kind of weight function. Popular choices are rectangular and Gaussian weights. All the experimental results presented in this paper use a standard rectangular IW ($w_{\mathrm{f}}(\boldsymbol{m})=1/(2R+1)^2$ for $\boldsymbol{m}\in\{-R,\ldots,R\}\times\{-R,\ldots,R\}$). For convenience of coding, we use only odd IW dimensions.

Now addressing the minimization process, let us assume that an initial guess $\boldsymbol{u}_0(\boldsymbol{k})$ of the displacement is available and is a good approximation of the sought displacement $\boldsymbol{u}(\boldsymbol{k})$, i.e., $\boldsymbol{u}(\boldsymbol{k})-\boldsymbol{u}_0(\boldsymbol{k})\approx 0$. The Gauss–Newton iteration derives from the following first-order expansion of Eq. 1 around $\boldsymbol{u}_0(\boldsymbol{k})$, with $\boldsymbol{u}(\boldsymbol{k})-\boldsymbol{u}_0(\boldsymbol{k})$ as a small parameter:

$$\sum_{\boldsymbol{m}} w(\boldsymbol{m}-\boldsymbol{k})\big(I_j(\boldsymbol{m},0)-I_j(\boldsymbol{m}-\boldsymbol{u}_0(\boldsymbol{k}),dt) + \nabla I_j(\boldsymbol{m}-\boldsymbol{u}_0(\boldsymbol{k}),dt)^t(\boldsymbol{u}(\boldsymbol{k})-\boldsymbol{u}_0(\boldsymbol{k}))\big)^2. \tag{3}$$

Equation 3 is a linear least-squares criterion, which can already be optimized to yield $\boldsymbol{u}(\boldsymbol{k})$ by solving a 2×2 linear system.

In Bouguet (2000), a faster scheme was proposed. It relies on a slightly different form of Eq. 1:

$$\sum_{\boldsymbol{n}} w(\boldsymbol{m}-\boldsymbol{k})\big(I_j(\boldsymbol{m}+\boldsymbol{u}(\boldsymbol{k})-\boldsymbol{u}_0(\boldsymbol{k}),0) -I_j(\boldsymbol{m}-\boldsymbol{u}_0(\boldsymbol{k}),dt)\big)^2. \tag{4}$$

In this criterion, instead of searching $\boldsymbol{u}(\boldsymbol{k})$ in the image at time dt as in Eq. 1, image at time dt is shifted by the estimate $\boldsymbol{u}_0(\boldsymbol{k})$ and the increment $\boldsymbol{u}(\boldsymbol{k})-\boldsymbol{u}_0(\boldsymbol{k})$ is applied to the image at time 0. A Taylor expansion of $I_j(\cdot,0)$ around $\boldsymbol{m}$ then yields

$$\sum_{\boldsymbol{m}} w(\boldsymbol{m}-\boldsymbol{k})\big(I_j(\boldsymbol{m},0)-I_j(\boldsymbol{m}-\boldsymbol{u}_0(\boldsymbol{k}),dt) + \nabla I_j(\boldsymbol{m},0)^t(\boldsymbol{u}(\boldsymbol{k})-\boldsymbol{u}_0(\boldsymbol{k}))\big)^2. \tag{5}$$

The advantage of this "inverse additive" approach (Baker and Matthews 2004) is that the spatial intensity gradient $\nabla I_j(\boldsymbol{m},0)$ is computed only once for each resolution level j, while in Eq. 3 the spatial gradient, $\nabla I_j(\boldsymbol{m}-\boldsymbol{u}_0(\boldsymbol{k}),dt)$ has to be computed at each iteration.

2.3 Dense LK algorithm

As shown in Le Besnerais and Champagnat (2005), if one wants to apply iterative techniques based on expansions (3) or (5) *at each pixel*—which is the usual goal in computer vision—the overall cost is prohibitive, because, for each iteration, it requires $(2R+1)^2$ interpolations per pixel due to the $I_j(\boldsymbol{m}-\boldsymbol{u}_0(\boldsymbol{k}),dt)$ term (and also because of the gradient $\nabla I_j(\boldsymbol{m}-\boldsymbol{u}_0(\boldsymbol{k}),dt)$ which appears in Eq. 3). Faster schemes can in fact be obtained by using only one interpolated image per iteration. To do so, we introduce the following notation:

$$I_j^{\boldsymbol{u}_0}(\boldsymbol{m},dt)\triangleq I_j(\boldsymbol{m}-\boldsymbol{u}_0(\boldsymbol{m}),dt). \tag{6}$$

As this expression shows, image $I^{\boldsymbol{u}_0}$ is "warped" according to the current displacement field estimate $\boldsymbol{u}_0$ *evaluated at each pixel* $\boldsymbol{m}$, as opposed to a warping with one value of $\boldsymbol{u}_0$ per IW. In order to obtain a convergent scheme based on the unique warped image (6), FOLKI thus uses the approximation proposed in Le Besnerais and Champagnat (2005):

$$\boldsymbol{u}(\boldsymbol{k})-\boldsymbol{u}_0(\boldsymbol{m})\approx 0,\quad \forall \boldsymbol{m}\in\mathcal{W}(\boldsymbol{k}) \tag{7}$$

Using Eq. 7, one then derives a first-order expansion of Eq. 1:

$$\sum_{\boldsymbol{m}} w(\boldsymbol{m}-\boldsymbol{k})\Big(I_j(\boldsymbol{m},0)-I_j^{\boldsymbol{u}_0}(\boldsymbol{m},dt) +\nabla I_j(\boldsymbol{m},0)^t(\boldsymbol{u}(\boldsymbol{k})-\boldsymbol{u}_0(\boldsymbol{m}))\Big)^2. \tag{8}$$

Minimization of Eq. 8 finally amounts to solving a 2×2 local system

$$\boldsymbol{H}(\boldsymbol{k})\boldsymbol{u}(\boldsymbol{k})=\boldsymbol{c}(\boldsymbol{k}). \tag{9}$$

Let us detail the computation of the matrices $\boldsymbol{H}(\boldsymbol{k})$ for all pixel index $\boldsymbol{k}$. While searching for the stationary point which minimizes Eq. 8, one obtains:

$$\boldsymbol{H}(\boldsymbol{k})=\sum_{\boldsymbol{m}} w(\boldsymbol{m}-\boldsymbol{k})\big(\nabla I_j(\boldsymbol{m},0)\nabla I_j(\boldsymbol{m},0)^t\big). \tag{10}$$

If $\boldsymbol{H}$ denotes the matrix valued function $\boldsymbol{k}\mapsto\boldsymbol{H}(\boldsymbol{k})$ of the pixel index, Eq. 10 for all pixels $\boldsymbol{k}$ can be globally written as a convolution:

$$\boldsymbol{H}=w*\big(\nabla I_j(\cdot,0)\nabla I_j(\cdot,0)^t\big), \tag{11}$$

where $*$ stands for the convolution of the scalar weight function w with each component of the matrix valued function which is inside the parenthesis. In the same way, the right-hand side vectors $\boldsymbol{c}(\boldsymbol{k})$ of Eq. 9 can be all computed by convolutions as follows

$$\boldsymbol{c}=w*(\epsilon\nabla I_j(\cdot,0)) \tag{12}$$

$$\epsilon=I_j(\cdot,0)-I_j^{\boldsymbol{u}_0}(\cdot,dt)-\nabla I_j(\cdot,0)^t\boldsymbol{u}_0 \tag{13}$$

As a result, at each iteration, local systems (9) for all pixels can be constructed *simultaneously* by Eqs. (6–11–13)—note however, that Eq. 11 can be computed once for all iterations, as already mentioned.

2.4 Overall algorithm and general comments

The global structure of the algorithm, summarized in Table 1, is a coarse-to-fine multiresolution scheme over J levels, with a fixed number N of Gauss–Newton iterations per level. As mentioned above, J should be chosen depending on the expected displacements in the image, and N may depend on the quality of the images and on the radius R of the IWs. More details on the way to choose these parameters will be given in Sects. 5 and 6. Also, note that the current version of FOLKI at use in ONERA gathers additional features specially adapted to PIV, which will be described in Sect. 4. Here, we simply comment on some specificities of the algorithm which are already contained in the above derivation.

A first remark is that, as shown in Table 1, each iteration begins with an image warp (6). Hence, FOLKI can be related to *image deformation techniques* (Lecordier and Trinite 2003; Stanislas et al. 2008). But, as FOLKI computes a dense vector field, the deformation is available at each pixel without velocity interpolation.

The dense character of the vector field also deserves further comment. First, it should be mentioned that it is an unavoidable building block of the algorithm: solving Eq. 9 for a restricted ensemble of spatial locations cannot be envisaged here, since computations (12, 13) require the availability of velocities at a much larger number of locations. Viewed in the PIV context, this by-product of the computer vision origin of FOLKI may however, appear useless, or even detrimental in terms of computational time. Indeed, as will be shown in Sect. 5, similarly to other PIV approaches based on window matching, FOLKI's spatial resolution remains related to the window size. A first important remark which justifies our choice is that this density is not an overload to the computational time: tests performed on a CPU implementation showed that the

Table 1 Pseudo GPU code of FOLKI

input: $I(\cdot,0)$ and $I(\cdot,dt)$
output: $\boldsymbol{u}=(u,v)$
begin
send $I(\cdot,0)$, $I(\cdot,dt)$ from CPU memory to GPU global memory
GPU: compute Burt pyramids *(SC)*
for $j=J-1:-1:0$
  GPU: compute $\nabla I_j(\cdot,0)$ *(SC)*
  GPU: compute $\boldsymbol{H}$ *(PW)*
  for $n=1:N$, **iterate:**
    GPU: compute $I_j^{\boldsymbol{u}_0}(\cdot,dt)$ *(II)*
    GPU: compute ϵ *(PW)*
    GPU: compute $\boldsymbol{c}$ *(PW+SC)*
    GPU: solve local systems (9) *(PW)*
  GPU: upsample $\boldsymbol{u}$ vector fields *(SC)*
(option 1) GPU: compute output image result and transfer it into GPU visualization memory
(option 2) send $\boldsymbol{u}=(u,v)$ result from GPU to CPU
end

GPU: GPU functions, *II*: image interpolation, *SC*: separable convolution, *PW*: pixelwise operation

computational time of a dense vector field with FOLKI was comparable to that of a classical sparse computation with a commercial PIV software. This is due to the high degree of optimization of FOLKI. Besides, and paradoxically, it is in fact this dense character that leads to a highly regular and parallel algorithm which precisely allows the considerable speed-up provided by the GPU. In addition, density provides an appreciable degree of freedom of result sampling, e.g., to finely evidence vortex cores or investigate flows close to walls, see Sect. 6.2 for an example.

3 Implementation on a GPU

An implementation has been developed in C++ and CUDA language for NVIDIA GPU and tested on different hardwares (generic graphic unit of a laptop, and a dedicated GPU on a PC workstation) with Linux and Windows OS.

Different packages of FOLKI are freely available on the ONERA website, at the address: http://www.onera.fr/dtim-en/gpu-for-image/index.php. Note that the open source Linux package strictly corresponds to the algorithm described in Sect. 2, whereas the Windows packages include the additional features described in Sect. 4.

The efficiency of the GPU implementation stems from the fact that FOLKI relies mainly on three types of computations, image interpolation (II), pixelwise operations (PW) and separable convolution (SC), see Table 1. These computations are performed very efficiently on a GPU, see Champagnat et al. (2009) for a more detailed account on GPU architecture and how to make profit of it. Two main features can be highlighted:

1. Image bilinear interpolation is hardwired on a GPU, it is thus performed at a cost which is negligible compared to a CPU.
   Higher order interpolation can also be performed very efficiently thanks to the algorithm of Ruijters et al. (2008) that combines multiple bilinear interpolations to perform one bicubic B-spline interpolation.
2. It is fundamental to limit the number of CPU-GPU transfer which are particularly time-consuming. The GPU pseudo-code presented in Table 1 is designed to minimize the number of CPU-GPU image transfers. Note the optional steps at the end of Table 1: if the code is used only to visualize an output image which depends on the computed velocity field (for instance an image of the vorticity field), it is much faster to compute this image with the GPU and then to transfer it directly into the visualization memory of the GPU. This mode can be very useful for fast parameter tuning of an experiment.

4 Adapting FOLKI to PIV context

We now discuss some extensions of FOLKI, directly dictated by the typical constraints of PIV experiments. These developments principally aim at increasing the accuracy, properly handling boundaries and giving the user a quality criterion on the obtained vector fields. Results using this improved version, which we call FOLKI-PIV, are presented in Sects. 5 and 6.

4.1 Third-order B-spline interpolator

As mentioned in Sect. 4, the user may choose whether the image interpolation is performed via simple bilinear interpolation, or using third-order B-splines. Having such a choice may prove relevant in order to adapt to the image characteristics, as will be shown for instance in Sect. 6.1.

4.2 Symmetric matching cost

Following symmetric SSD criteria like

$$\sum_{\boldsymbol{m}} w(\boldsymbol{m}-\boldsymbol{k})\left(I_j\left(\boldsymbol{m}+\frac{\boldsymbol{u}(\boldsymbol{k})}{2},0\right)-I_j\left(\boldsymbol{m}-\frac{\boldsymbol{u}(\boldsymbol{k})}{2},dt\right)\right)^2, \tag{14}$$

have been proposed by many authors (Keller and Averbuch 2004; Zhao and Sawhney 2002), in order to suppress the dissymmetry of classical SSD costs, increase precision and robustness against occlusions. When upgrading toward FOLKI-PIV, we chose to implement this approach rather than the simple original SSD criterion (1).

In practice, Eq. 14 can be handled in a very similar manner as Eq. 1. Replace $u(\boldsymbol{k})$ by $u_0(\boldsymbol{m})+(u(\boldsymbol{k})-u_0(\boldsymbol{m}))$ in Eq. 14, then take first-order approximation for both images based on Eq. 7. Finally, one gets modified expressions for Eqs. 11, 12 and 13. For instance, the 2×2 matrix $\boldsymbol{H}(\boldsymbol{k})$ associated to pixel $\boldsymbol{k}$ now writes:

$$\sum_{\boldsymbol{m}} w(\boldsymbol{m}-\boldsymbol{k})\left(\nabla I_j\left(\boldsymbol{m}-\frac{\boldsymbol{u}_0(\boldsymbol{m})}{2}\right)\nabla I_j\left(\boldsymbol{m}-\frac{\boldsymbol{u}_0(\boldsymbol{m})}{2}\right)^t + \nabla I_j\left(\boldsymbol{m}+\frac{\boldsymbol{u}_0(\boldsymbol{m})}{2},dt\right)\nabla I_j\left(\boldsymbol{m}+\frac{\boldsymbol{u}_0(\boldsymbol{m})}{2},dt\right)^t\right), \tag{15}$$

(the '0' in $\nabla I_j(.,0)$ has been omitted for concision). The overall structure of the symmetric algorithm remains similar to the one in Table 1, except that expression (15) has to be recomputed at each iteration using spatial gradients and interpolations of both images. In this process, the computational time is multiplied by 2 compared to the basic algorithm of Sect. 2.

4.3 Robustness to varying illumination

In contrast to a zero-normalized cross-correlation (ZNCC) maximization objective, which is classically used in PIV, objectives defined by SSD minimization such as Eq. 14 are less robust to varying illumination conditions. Of course, global illumination changes can easily be handled by equalization using image gain and offset adjustment before feeding the algorithm with the corrected image pairs. But this approach will not work when, for instance, the lighting difference varies across the field of view, a situation which is encountered in PIV, see for instance both examples of Sect. 6. In this case, some kind of local equalization is required.

We follow hereafter the logic of mean and standard deviation normalization; note that the min–max normalization of Westerweel (1993) could also be implemented cheaply on a GPU. The principle of such a local equalization is to replace the image intensity values $I(\cdot,t)$ $(t = \{0, dt\})$ by normalized intensities $\widetilde{I}(\cdot,t)$. For an IW centered on a pixel $\boldsymbol{k}$, the vector of normalized intensities of pixels $\boldsymbol{m}$ inside the IW, $\{\widetilde{I}(\boldsymbol{m},t)\}_{\boldsymbol{m}\in\mathcal{W}(\boldsymbol{k})}$, should have approximately zero mean and unit standard deviation. If the displacement field is zero, such a normalization simply writes

$$\widetilde{I}(\boldsymbol{m},t) = (I(\boldsymbol{m},t) - M(\boldsymbol{k},t))/\sigma(\boldsymbol{k},t), \quad t = \{0, dt\},$$

where the local empirical mean $M(\cdot,t)$ and standard deviation $\sigma(\cdot,t)$ are computed simultaneously for all pixels by pixelwise operations and separable convolution.

For each iteration of FOLKI-PIV, the current displacement field is not zero anymore and local means and standard deviation should be computed on the warped images $I^{-\boldsymbol{u}_0/2}(\cdot,0)$ and $I^{\boldsymbol{u}_0/2}(\cdot,dt)$ (using the notation from Eq. 6), in order to write a normalized symmetric criterion from Eq. 14. A fast approximation is to perform the normalization only once, at the beginning of each level, and then performing the GN iterations on these images. With such a strategy, there is nearly no extra cost and empirical comparisons of both schemes—exact or approximate—show their equal effectiveness.

4.4 Boundary handling

This problem should be adequately addressed not only to process pixels located near the boundary of the field of view but also to take masks into account. A mask is a binary image aimed at excluding some pixels from the estimation process, because they belong to some rigid object (wing, measurement device, etc.) present in the field of view; see for instance the real dataset of Sect. 6.2. In the sequel, the image support refers to the set of non-masked pixels.

It is quite delicate to find an optimal way to handle boundaries in window-based displacement estimation. Indeed the *support of the estimation*, i.e., the pixels for which the system (9) can be constructed and inverted, varies with the estimate: if the displacement field locally tends to escape from the image, the support of the estimation consequently "moves back" away from the boundary. Such effect is even more pronounced in FOLKI, because of the dense estimation and also of the multiresolution process.

The proposed solution retained in FOLKI-PIV relies on dynamic masks which are updated at each iteration. Excluded pixels are (*i*) those whose current displacement vector falls outside of the image support (in the symmetric case, the displacement fields which are used are either $\boldsymbol{u}_0/2$ or $-\boldsymbol{u}_0/2$); (*ii*) those whose IW contains more than 80% of already excluded pixels. Displacement vectors are computed for the remaining pixels, then the holes are filled with nearest valid vectors before the next iteration.

4.5 Correlation quality

When analyzing PIV images, imperfect lighting, particle loss, and CCD noise will impact the quality of the correlation and thus increase the uncertainty of a computed vector. To yield a quality criterion to the user, we included in FOLKI-PIV a computation of the correlation peak height, as is traditionally done in PIV. This is done by first warping the images by the final displacement, so as to get images $I^{-\boldsymbol{u}/2}(\cdot,0)$ and $I^{\boldsymbol{u}/2}(\cdot,dt)$. Then, mean and standard deviation normalization is applied on these images, yielding $\widetilde{I}^{-\boldsymbol{u}/2}(\cdot,0)$ and $\widetilde{I}^{\boldsymbol{u}/2}(\cdot,dt)$. As a preliminary step, the ZNSSD score S_{ZNSSD} is then computed. This quantity is simply the residual of the symmetric SSD criterion (14) applied to these zero- normalized images:

$$S_{\text{ZNSSD}} = \sum_{\boldsymbol{m}} w(\boldsymbol{m}-\boldsymbol{k})\left(\widetilde{I}_0^{-\boldsymbol{u}/2}(\boldsymbol{m},0) - \widetilde{I}_0^{\boldsymbol{u}/2}(\boldsymbol{m},dt)\right)^2. \tag{16}$$

It finally turns out that S_{ZNSSD} is directly related to the classical correlation score S_{ZNCC}, or height of the correlation peak, which is defined as

$$S_{\text{ZNCC}} = \sum_{\boldsymbol{m}} w(\boldsymbol{m}-\boldsymbol{k})\widetilde{I}_0^{-\boldsymbol{u}/2}(\boldsymbol{m},0)\widetilde{I}_0^{\boldsymbol{u}/2}(\boldsymbol{m},dt). \tag{17}$$

Simple algebra, see for instance Pan et al. (2007, Appendix A), indeed shows that one has

$$S_{\text{ZNCC}} = 1 - \frac{S_{\text{ZNSSD}}}{2}. \tag{18}$$

Consequently, even though FOLKI-PIV's objective is formulated differently as in classical PIV, the quality of its results

may be assessed—and vectors validated or not—in the same way, using S_{ZNCC}. $S_{\mathrm{ZNCC}} = 1$ will indicate a perfect matching, while $S_{\mathrm{ZNCC}} = 0$ is the theoretical limit of no correlation. Further elements on the way we use this score in practice will be given in Sects. 5 and 6.

5 Performance assessment: synthetic data

In the following, we use synthetic PIV images to study FOLKI-PIV's spatial frequency response, along with the effects of low seeding and noise on the reliability of the results. In order to determine whether the computation choices underlying FOLKI-PIV result in a different behavior as traditional PIV algorithms, we first present simple test cases for which the behavior of these algorithms is already documented. Then, we provide comparison of FOLKI-PIV's result to that of a state-of-the-art commercial software, hereafter denoted CPIV.

In the following, CPIV's estimation of an IW's displacement relies on the FFT-based computation of the cross-correlation map, followed by a Gaussian sub-pixel interpolation of the correlation peak. This process is embedded in an adaptive multi-pass scheme, in which the displacements are progressively refined from their previous estimates; this can be done with IWs of gradually decreasing size. Between each pass, rejection of spurious vectors is performed using a median filter, outliers being replaced either by other correlation maxima or by local interpolation, and the vector field is filtered with a 3×3 pixels Gaussian.

5.1 Parameters for the tests

The synthetic images used in the present work were generated with the EUROPIV Synthetic Image Generator (S.I.G.) which is described in Lecordier and Westerweel (2003). Keeping physical units in pixels, we use $1{,}025 \times 1{,}025$ images with a fully covered 8 bit range. Particles show as 2 pixels diameter Gaussian intensity distribution. The intensity level of a given particle depends on its out-of-plane position with respect to the $l_w = 2$ pixels width Gaussian-shaped light sheet that illuminates the scene. If N_p is the total number of particles in a volume V illuminated by the laser sheet, then particle density seen in the image is $N_d = \frac{N_p l_w}{V}$. Unless specified otherwise, particle density is set to $N_d = 0.02$ and no CCD noise is added, yielding sample images such as shown in Fig. 2. Displacement fields are applied symmetrically forward and backward to an initial cloud of randomly located particles. For each test case, identical displacement patterns are repeated in different zones of a given image pair, so that 25 image pairs are sufficient to achieve statistical convergence of the results.

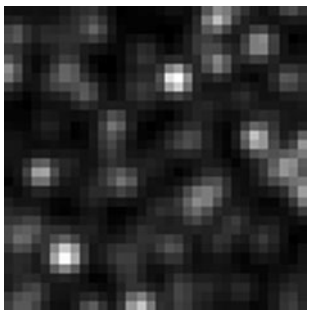

Fig. 2 32×32 pixel close-up at a S.I.G. image with $N_d = 0.02$

FOLKI-PIV's window radius R is varied from 5 to 31 pixels, and the interrogation window has a standard rectangular weight. We use $J = 1$ level because the pixel displacements in these tests are less than 2 pixels. Convergence is reached for $N = 3$ iterations. The interpolation is performed with a third-order B-spline.

CPIV is also used with various IW sizes (8×8, 16×16, 32×32 and 64×64 pixels) with rectangular weight. For a given result, the calculation is done thanks to a multi-pass scheme composed of 4 passes with the same IW size. We use a Whittaker pixel interpolation method and overlap is set to 75%.

5.2 Spatial frequency response

Following Scarano and Riethmuller (2000), the frequency response of a PIV algorithm can be evaluated using a sinusoidal shear displacement test:

$$(U, V) = \left(A \sin\left(2\pi \frac{Y}{\lambda} \right), 0 \right) \tag{19}$$

where X and Y are the horizontal and vertical coordinates and U and V the associated displacement components.

We here reproduce this displacement field with amplitude A set to 2 pixel, and the wavelength λ varied from 20 to 400 pixels. Figure 3 compares the ground truth and the average value found by FOLKI-PIV for the U component, for case $\lambda = 200$ processed with $R = 10$ IWs. For each image corresponding to a given (λ, R) couple, we computed the ratio between the estimated amplitude of the sinusoid $A_{\mathrm{FOLKI-PIV}}$ and the ground truth value $A = 2$ pixel. The evolution of this ratio as a function of the normalized IW size $2R/\lambda$ is plotted in Fig. 4. Although we used IW radii ranging from $R = 5$ to $R = 31$, all values of $A_{\mathrm{FOLKI-PIV}}/A$ nearly collapse on a cardinal sine curve, which is the frequency response of a $[-R, R]$ sliding average. In comparison, as shown by Scarano and Riethmuller (2000), iterative multigrid methods with isotropically weighted

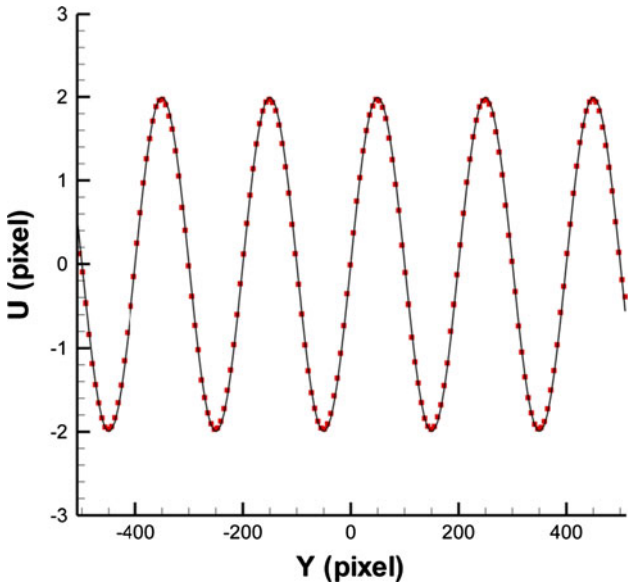

Fig. 3 Sinusoidal shear displacement with $A = 2$ pixels and $\lambda = 200$ pixels. Ground truth (*black curve*) and average displacement found by FOLKI-PIV with $R = 10$ IWs, downsampled every 6 pixels

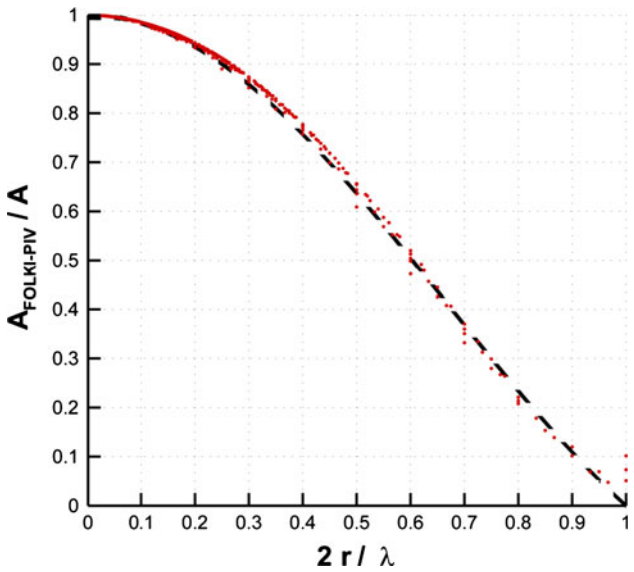

Fig. 4 Sinusoidal shear displacement with $A = 2$ pixels: amplitude ratio $A_{\text{FOLKI-PIV}}/A$ as a function of the normalized window size $2R/\lambda$ (*red symbols*), for IW radii R ranging from 5 to 31. The *dashed line* is the response of a $[-R, R]$ sliding average (cardinal sine function)

IWs follow a similar trend, but with an amplitude damping compared to the ideal sliding average, which we do not observe with FOLKI-PIV.

5.3 Resolution versus noise

One other way to evaluate the effective spatial resolution is to recover the response of the algorithm to a sharp spatial

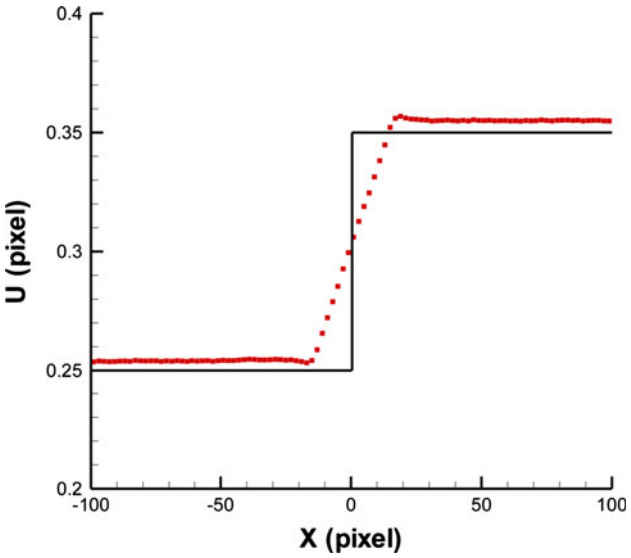

Fig. 5 Sharp horizontal step: displacement estimated by FOLKI-PIV with $R = 15$ (*red dots*, downsampling every 2 pixels) and ground truth (*black curve*). Note that the displacement error for FOLKI-PIV which appears from this figure (roughly 0.005 pixel) remains well below the usual PIV uncertainty

step in the displacement field, similar to the high velocity gradient that can be found across a shock wave. The following test was suggested to us by B. Wieneke (private communication): the displacement field is a sudden step from a $U = 0.25$ to a $U = 0.35$ pixel horizontal translation. Figure 5 compares the average result found for U by FOLKI-PIV with $R = 15$ to the ground truth. For a given window radius R, we determine the effective spatial resolution by measuring the width over which the PIV algorithm integrates the sharp edge. In practice, this width can be recovered as the inverse of the slope of the estimated $U(X)$ at the center of the step. Figure 6, in which this quantity is plotted against R, shows that FOLKI-PIV integrates the step exactly like the $[-R, R]$ moving average, which has a $2R$ effective spatial resolution.

An experimentalist seeking a better spatial resolution will be tempted to use smaller window sizes, but the drawback is an increased measurement uncertainty. Previous studies have shown that this resolution versus uncertainty trade-off also depends on the quality of the images (Raffel et al. 2007, pp. 174–176). To show how FOLKI-PIV behaves in that respect, we have considered three test cases keeping the same sudden step displacement, each of them characterized with a different type of image degradation or added noise compared to the above ideal situation. For each of them, we compute the rms displacement error U_{RMS} of both FOLKI-PIV and CPIV, for various IW sizes.

First, we have added an out-of-plane displacement $W = 0.5$ pixels so that some particles are lost (case I). W is to be compared to the Gaussian light sheet width $l_w = 2$

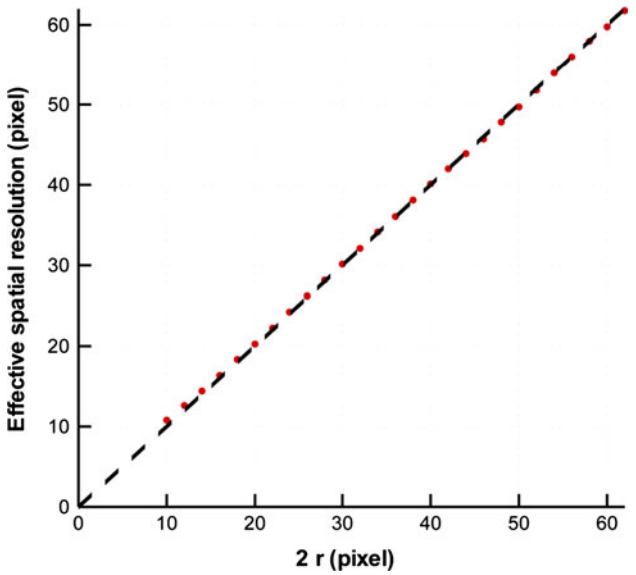

Fig. 6 Effective spatial resolution of FOLKI-PIV as a function of the total IW width $2r$ (*red symbols*) computed from the sharp horizontal step test case, compared with the effective spatial resolution of the $[-R, R]$ sliding average (*dashed black curve*)

pixels. Second, we have added to the images a Gaussian CCD noise with 20 gray level standard deviation, while reducing the 8 bit dynamic range down to 7 bit, thus reducing the signal to noise ratio (case II). Finally, using the same CCD noise as in case II, we have reduced the seeding density from $N_d = 0.02$ down to $N_d = 0.005$ (case III). In Fig. 7, FOLKI-PIV's rms displacement error U_{RMS} is compared to CPIV's for different IW sizes. For all these

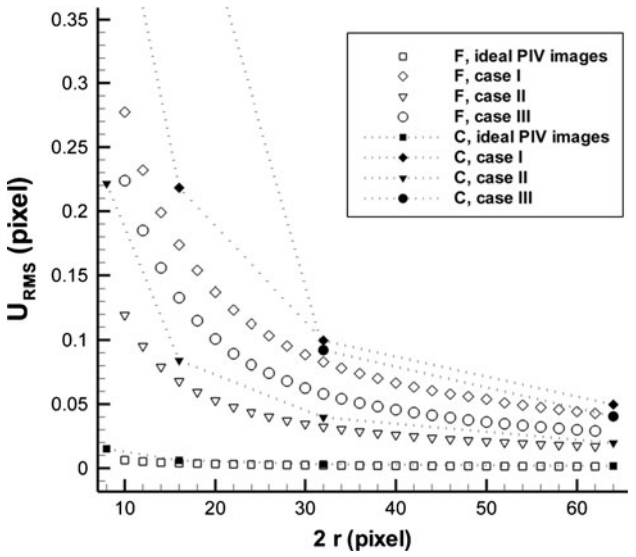

Fig. 7 Rms displacement error U_{RMS} against IW size $2R$ for three test cases of sharp horizontal step displacement (see the text for their exact characteristics). FOLKI-PIV (F) and CPIV (C) results in open and filled symbols, respectively. For clarity, CPIV results are linked by *dotted lines*

cases, FOLKI-PIV and CPIV have a similar behavior, the rms error rising as the IW size decreases. But remarkably, it turns out that FOLKI-PIV performs equally or better than CPIV depending on the cases, whereas it does not involve any data post-processing between two successive iterations, as CPIV does. This is especially true for small windows sizes.

The results presented in this section show a reassuring behavior of FOLKI-PIV with respect to noise and resolution, in other terms FOLKI-PIV does not sacrifice measurement accuracy for speed.

6 Performance assessment: real data

In the following, we provide results on two experimental PIV datasets using FOLKI-PIV with the improvements described in previous sections. The main purposes of these tests are to refine the assessment of FOLKI-PIV's rms displacement error for difficult experimental conditions, as well as its peak-locking bias error, to show how FOLKI-PIV deals with solid boundaries and illustrates the advantages of density for result sampling.

6.1 Case A of the second PIV challenge

Our first real dataset is case A of the second PIV challenge (Stanislas et al. 2005). In this experiment, a round turbulent jet is imaged at a distance from its exhaust large enough for the flow to be self-similar there. This way, a quantitative performance assessment is made possible, by comparison with canonical self-similar turbulent jets, even though the data are extracted from a real experiment with typically encountered difficulties. The sample retained in the 2003 PIV challenge consists of 100 images of resolution 992 × 1,004 pixels, in which the particles images have a diameter estimated to 1 pixel. As described by Stanislas et al. (2005), the main goals of proposing this test case were to assess the ability of PIV algorithms to deal with strongly turbulent flows, as well as to evaluate their rms displacement error and their peak-locking error.

6.1.1 Choice of the vector computation parameters

In the following, as done in the previous paragraph, we compare the results obtained by both FOLKI-PIV and CPIV on these data. For the former, we use $J = 3$ levels, $N = 7$ iterations and $R = 15$ IWs, and for the latter, a multi-pass iterative scheme composed of 2 passes with 64 × 64 IWs followed by 4 passes with 32 × 32 IWs (50% overlap at each pass), with the same intermediate spurious vector rejection and data processing between passes as described in Sect. 5. After the last pass of CPIV, this post-processing is applied once again to yield the final vector fields. For both

algorithms, local zero normalization of the image intensities is applied, and a bilinear interpolator is used, instead of the more accurate interpolators used in the previous paragraph. It is indeed known that for data having a low signal to noise ratio, bilinear interpolation yields optimal performance, especially regarding peak-locking (Lecordier and Trinité 2006; Yamamoto and Uemura 2009). Our tests (not shown here for conciseness) confirmed this conclusion for the present dataset.

Concerning FOLKI-PIV's settings, the choice $N = 7$ stems from a convergence study performed on two image pairs, labeled 1 and 5 in the dataset. We processed these images with N increasing from 1 to 20, keeping all other parameters constant. For each value of $N \geq 2$, we here compute at each image point (X, Y) the norm $\Delta(X, Y)$ of the difference between the displacement vectors found at iterations N and $N - 1$. The optimal choice for the iteration number then results from the fact that for $N \geq 7$, Δ is observed to be inferior to 0.1 pixel in the whole field, this limit being a traditional estimate of PIV accuracy on experimental images. Figure 8 below yields an illustration of this convergence, by showing the evolution with N of Δ, together with the correlation score S_{ZNCC} at $(X, Y) = (497, 502)$. As can be observed, at this location, an accurate result is reached for N well below 7. More importantly, the correlation score convergence is seen to be similar to that of Δ. This shows that the user may tune the parameters of FOLKI-PIV by a visual diagnosis of either the velocity fields or the correlation score S_{ZNCC}, by monitoring when the fields cease to change as N is increased. In practice, this quick preliminary convergence study should be led bearing the following typical values in mind: good quality images usually yield converged results for N as low as 3 (as was the case for the synthetic images of Sect. 5), whereas experimental images of poor quality may require values of N reaching up to 10.

6.1.2 Comparative results

To compare FOLKI-PIV's and CPIV's results, we downsample the dense vector fields of FOLKI-PIV every 15 × 15 pixels. Upon applying for both datasets the same a posteriori detection of outliers as in Stanislas et al. (2005), which amounts to keeping vectors lying in the inside of a given ellipse in a (U, V) scatter plot, we found 5 outliers with FOLKI-PIV, over a total of 442,200 vectors. In comparison, 87 vectors were found outside of the ellipse with CPIV, over a total of 390,600. It is worth noticing that FOLKI-PIV yields a remarkably small number of outliers despite the fact that it does not include any spurious vector rejection step between iterations—whereas CPIV does.

Results for the peak-locking bias are shown in Figs. 9 and 10, where the global histogram for U and the histogram for the fractional part of U are plotted, respectively. These curves show that for images with particles of small size, FOLKI-PIV displays a level of peak-locking bias comparable with that of CPIV, with a slightly higher level for even values of the displacement. To date, we have no explanation for this difference, as both algorithms use a symmetrical window shift before each iteration or pass. Overall, both figures thus show that, even though it does not involve any interpolation of the correlation peak, FOLKI-PIV performs comparably to algorithms based on multi-pass schemes involving a three-point Gaussian

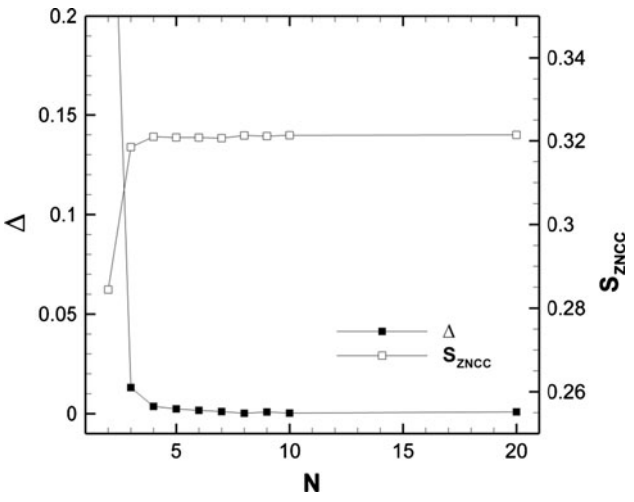

Fig. 8 Convergence with N of the displacement found by FOLKI-PIV at $X = 497$, $Y = 502$, image 1 of PIV challenge 2003 A's dataset. Plotted quantities are the correlation score S_{ZNCC} and the norm $\Delta(X, Y)$ of the difference between the displacement vector at iterations N and $N - 1$

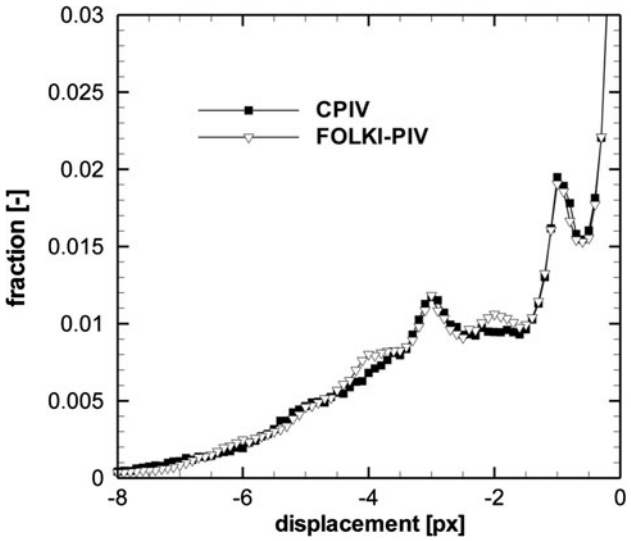

Fig. 9 Histograms of the U displacement

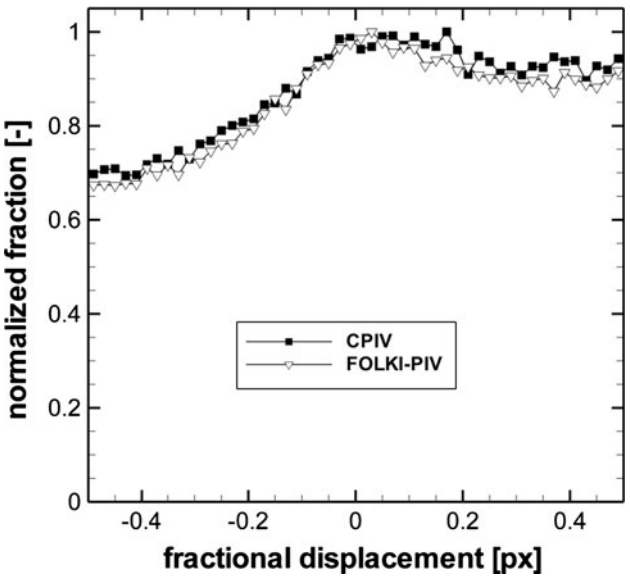

Fig. 10 Fractional part of the U displacement

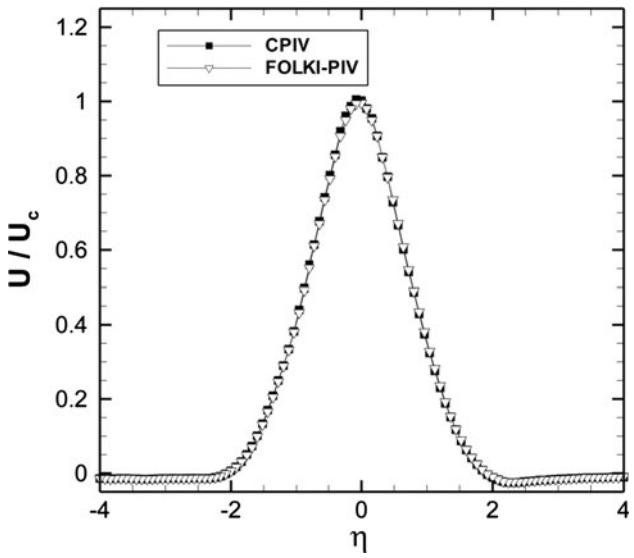

Fig. 11 Normalized mean horizontal displacement U/U_c as a function of the self-similar variable η

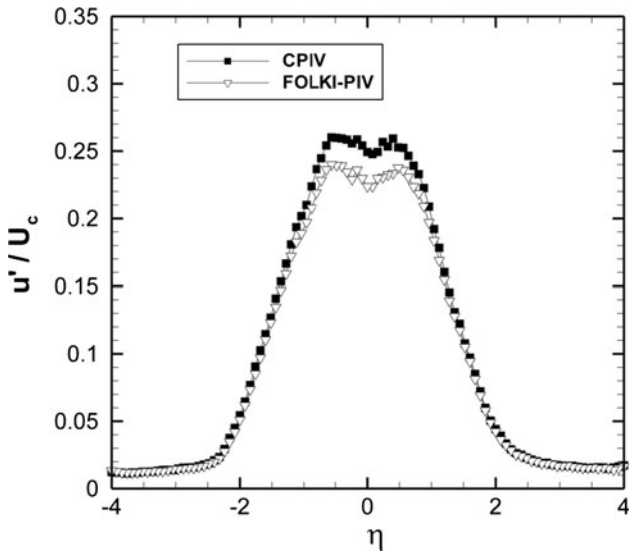

Fig. 12 Normalized rms horizontal displacement u'/U_c as a function of the self-similar variable η

sub-pixel interpolation (Westerweel 2000a, b; Westerweel et al. 1997).

Still following Stanislas et al. (2005), we now represent the mean and rms axial velocities U and u' as single one-dimensional profiles. This can be achieved by taking advantage of the jet self-similarity: first, the ensemble averaged axial velocity U must be fitted to

$$U(x-x_0, y-y_0) = U_c(x)\exp\left(-\eta^2\right) = U_c(x)\exp\left(-\frac{y^2}{l(x)^2}\right) \quad (20)$$

with

$$U_c(x) = \frac{A}{x-x_0} \quad \text{and} \quad l(x) = B(x-x_0), \quad (21)$$

using parameters B (jet spreading rate), x_0 and y_0 (jet virtual origin), and A. Then, values of U/U_c and u'/U_c from each profile at a given x may be gathered so as to represent one single profile depending on η. This is done in Figs. 11 and 12, respectively.

Concerning the mean velocity, an excellent agreement is found between CPIV and FOLKI-PIV. Both curves collapse almost perfectly, both in the jet itself and in the outer flow. The same remark also holds for the rms u' in the outer flow and in the jet shear layers (see Fig. 12). In the middle of the jet, for $-1.5 \leq \eta \leq 1.5$, discrepancies between both softwares are however observed, which are maximal in the plateau region, for $-0.6 \leq \eta \leq 0.6$. The average plateau value found by CPIV is of roughly 0.255, while that found by FOLKI-PIV is of roughly 0.235. Note that discrepancies of the same order of magnitude were found between algorithms tested during the second PIV challenge. To us, this is not surprising, since the present test case corresponds to difficult experimental conditions (seeding inhomogeneities in the individual images, differences in light intensity between two images in a pair, small particles). In practice, we find quite low values of the correlation score S_{ZNCC} close to the jet centerline, especially near the right edge of the images (typically, between 0.1 and 0.3). This may explain why algorithms may perform differently there.

As a last quality assessment from this test case, we consider the level of u'/U_c obtained in the outer flow region, where turbulence should progressively return to zero when $|\eta|$ increases. Similarly to the analysis performed

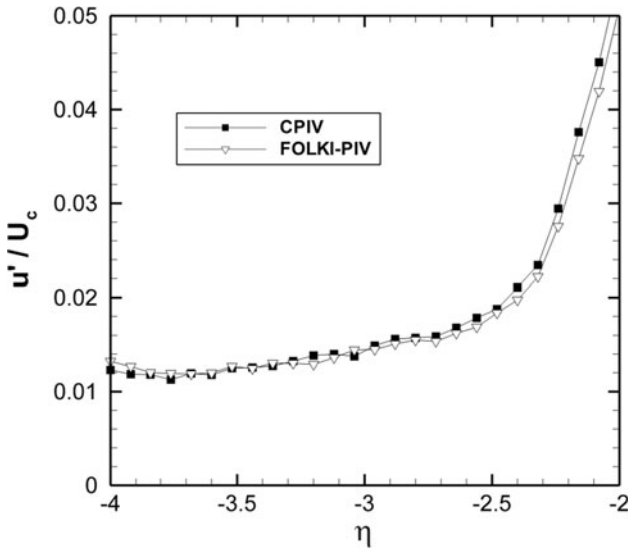

Fig. 13 Normalized rms horizontal displacement u'/U_c as a function of η, close-up

Table 2 Per image average computational time (in s), PIV challenge 2003 A dataset

FOLKI-PIV	CPIV standard	CPIV-high accuracy
0.2	9.4	19.4

All computations were performed on a PC with an Intel Core2 CPU at 3.16 *GHz*, 3 *Go* RAM and a NVIDIA Tesla C1060 GPU. See the text for more details on the settings used for each algorithm

by Stanislas et al. (2005), we plot in Fig. 13 a close-up of u'/U_c in the region $-4 \leq \eta \leq -2$. For $-4 \leq \eta \leq -3.5$, both codes reach approximately the same asymptotic limit of $u'/U_c \approx 0.012-0.013$. There as well, it appears that FOLKI-PIV performs nearly identically to a state-of-the-art PIV software using a FFT-based cross-correlation and a three-point Gaussian sub-pixel interpolation (Westerweel 2000a, b; Westerweel et al. 1997).

Finally, we compare in Table 2 the average time of computation of one vector field of this dataset for each algorithm. For CPIV, we also performed a simpler computation, composed of 1 pass with 64 × 64 IWs followed by 2 passes with 32 × 32 IWs, still with 50% overlap. Such a setting is usually a standard for a majority of PIV images, leading to a high enough accuracy. In Table 2, it is denoted by "CPIV-standard", as opposed to the setting used for the above comparisons, which is denoted by "CPIV-high accuracy". All these computations were performed on a machine equipped with an Intel Core2 CPU at 3.16 *GHz*, 3 *Go* RAM and a NVIDIA Tesla C1060 GPU.

As shown by Table 2, the high level of optimization of FOLKI together with the GPU implementation leads to computational gains ranging from 50 to 100 compared to a state-of-the-art traditional PIV approach, with a similar accuracy of the results.

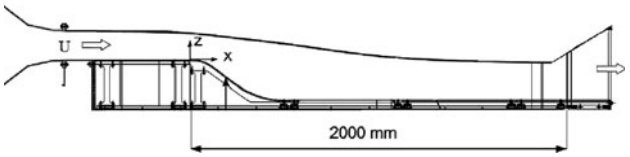

Fig. 14 Principle sketch of the ONERA S19Ch wind-tunnel

6.2 Massive flow separation over a rounded ramp (ONERA data, Gardarin et al. 2008)

The data considered in this paragraph are extracted a from High-Speed PIV (HS-PIV) test campaign on massive boundary layer separation and its control Gardarin et al. (2008), performed in the S19Ch wind-tunnel of ONERA Meudon center (see Fig. 14). In this closed-loop wind-tunnel, the test section is 300 mm wide and 2 m long and allows convenient optical access thanks to transparent lateral walls. As seen in Fig. 14, the flow enters the test section with a horizontal velocity $U = 30$ m s^{-1}. Separation then occurs due to the strong adverse pressure gradient created by the rounded ramp. While the purpose of the global test campaign has been to evaluate the efficiency of several control devices [in particular mechanical and fluidic vortex generators, Gardarin et al. (2008)], we here present the reference case, where the unforced separation occurs.

Figure 15 shows a sample image obtained with a HS-PIV system composed of a Litron LDY 303HE25 Nd:YLF laser and a Phantom V12.1 camera (1,280 × 800 pixels CCD). The generated laser sheet is 1 mm thick, and the flow is seeded using DiEthylHexyl Sebacate (DEHS) droplets. The repetition rate of the laser is set to 3 kHz, and the time interval between two images in a pair is 60 μs. The zone of interest is a rectangular field located near the

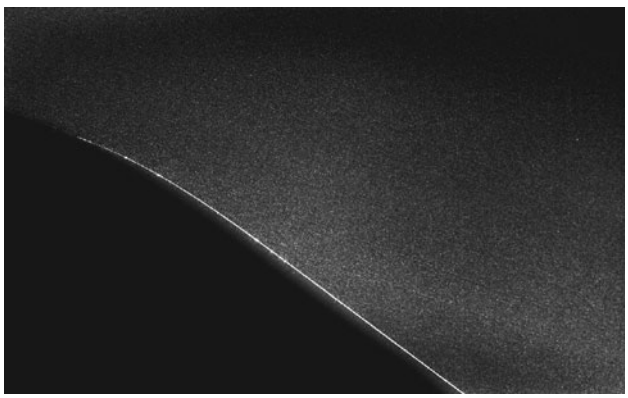

Fig. 15 PIV image sample of ONERA S19Ch HS-PIV experimental dataset

beginning of the rounded ramp, where boundary layer separation occurs. As mentioned above, this test case is specially interesting in that it combines several common experimental difficulties: a solid wall (the rounded ramp) is included inside the field of view so that a light reflection occurs on it, and the light intensity is not homogeneous in the bulk of the image, with in particular irregularities in the light sheet.

We have processed this PIV image pair with FOLKI-PIV using mean and standard deviation local normalization, and a mask following the contour of the rounded ramp. We used the following parameters: $R = 16$ IWs, $N = 6$ iterations, $J = 3$ pyramid levels. The value of N was chosen after a qualitative convergence study in the same way as described in Sect. 6.1.1. Results are shown in Figs. 16 and 17, which represent the obtained velocity field, together with vorticity iso-contours post-processed from this velocity field. To compute the velocity gradients involved in the vorticity, we used a 5×5 Gaussian filter before applying a traditional second-order finite difference scheme. Note that in both figures, the velocity field has been downsampled, respectively, to every 32×32 and 6×6 pixels.

Figure 16 shows that despite the difficult configuration, the flow physics is fully retrieved, since the boundary layer separation is precisely captured (here it is visible at around $x = -0.055$), as well as the various vortical structures which develop on the downstream mixing layer. Figure 17 confirms that FOLKI-PIV and its improvements described in Sect. 4 yield physically sound results even close to the wall, as seen from the various recirculation vortices which are captured until the mask edge. In this respect, it is important to emphasize that the dense result provided by the improved FOLKI-PIV is a real advantage, since it allows to sample the vector field even very close to obstacles.

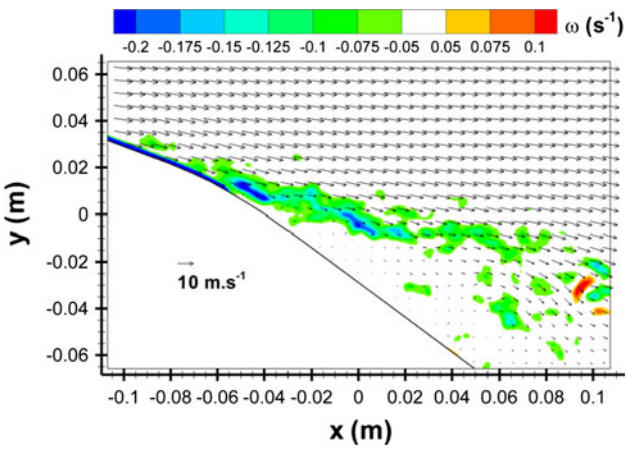

Fig. 16 Instantaneous velocity field obtained with FOLKI-PIV, together with iso-contours of vorticity post-processed from the velocity field by using a 5×5 pixels Gaussian filter. Vectors are displayed every 32×32 pixels

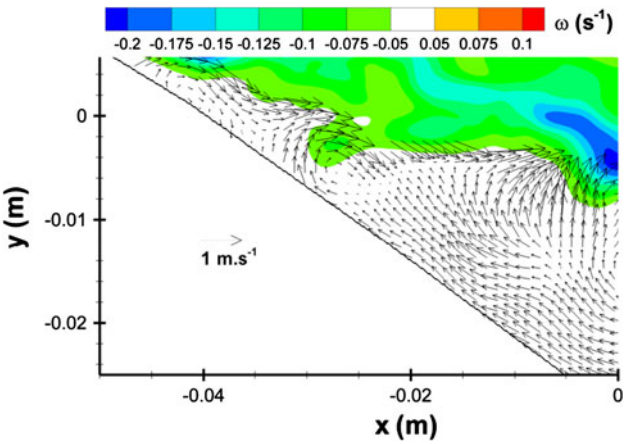

Fig. 17 Instantaneous velocity and vorticity field, close-up on the recirculation region. For clarity, we only plot velocity vectors which have a modulus inferior to 1 m s^{-1}. These vectors are displayed every 6×6 pixels

7 Conclusions

We have presented FOLKI-PIV an algorithm that performs fast computation of dense two-component (2C) PIV vector fields using Graphics Processing Units. Tested on both synthetic and real images, FOLKI-PIV proved as accurate as a state-of-the-art standard PIV software, while being 50 times faster and allowing a convenient degree of freedom for result sampling due to its dense character.

Our simulation study confirms that the behavior of FOLKI-PIV is that of a moving average filter—with $1/(2R)$ bandwidth, R being the window's radius. Robustness to noise is equal or better than state-of-the-art PIV software, without the need for any spurious vector rejection process between iterations. As a result, FOLKI-PIV appears very simple to tune for the practitioner.

To date, these features of FOLKI-PIV already offer to the experimentalists at ONERA a precious help in the way to conduct their PIV experiments, by allowing faster diagnosis on their experimental settings. Indeed, the accuracy of optical parameter tunings may now in particular be assessed also by investigating the mean and rms of the flows. This is especially precious when dealing with turbulent flows, which require a large number of image pairs for the turbulent statistics to converge. More generally, we believe that such a fast PIV code opens the way to significant changes in the way to perform experiments in fluids with PIV, allowing a much faster flow diagnosis and extending considerably possibilities of trial-and-error in the optimization phasis of experimental campaigns.

The basic rationale of the algorithm is furthermore generic enough to enable various extensions. On-going work considers Stereo PIV [see Leclaire et al. (2010)] and 3D velocity estimation for tomographic PIV, both settings which can be cast into the FOLKI-PIV scheme, so as to derive fast GPU implementation in the near future.

Acknowledgments Benoît Gardarin, Laurent Jacquin and Gilles Losfeld are gratefully acknowledged for providing their experimental TR-PIV dataset. Moreover the authors salute the helpful comments of the referees.

References

Baker S, Matthews I (2004) Lucas-Kanade 20 years on: a unifying framework. Int J Comput Vis 56(3):221–255

Bergen JR, Anandan P, Hanna KJ, Hingorani R (1992) Hierarchical model-based motion estimation. In: ECCV 92, Proceedings of the second European conference on computer vision, pp 237–252

Bouguet JY (2000) Pyramidal implementation of the Lucas-Kanade feature tracker: description of the algorithm. Tech. rep., Open CV Documentation

Burt P, Adelson E (1983) The Laplacian image pyramid as a compact image code. IEEE Trans Commun 31:532–540

Champagnat F, Plyer A, Le Besnerais G, Leclaire B, Le Sant Y (2009) How to calculate dense PIV vector fields at video rates. In: Proceedings of 8th international symposium on particle image velocimetry—PIV09, Melbourne

Corpetti T, Heitz D, Arroyo G, Mémin E, Santa-Cruz A (2006) Fluid experimental flow estimation based on an optical-flow scheme. Exp Fluids 40(1):80–97

Gardarin B, Jacquin L, Geffroy P (2008) Flow separation control with vortex generators. In: AIAA 4th flow control conference, Seattle, WA, 23–26 June

Iriarte Munoz J, Dellavale D, Sonnaillon M, Bonetto F (2009) Real-time particle image velocimetry based on FPGA technology. In: Programmable logic, 2009. SPL. 5th Southern conference on, pp 147–152

Keller Y, Averbuch A (2004) Fast motion estimation using bi-directional gradient methods. IEEE Trans Image Process 13(8):1042–1054

Le Besnerais G, Champagnat F (2005) Dense optical flow by iterative local window registration. In: ICIP'05, Proceedings of the IEEE international conference on image processing, vol I, pp 137–140

Leclaire B, Le Sant Y, Davoust S, Le Besnerais G, Champagnat F (2010) FOLKI-SPIV: a new, ultra-fast approach for stereo PIV. submitted to Experiments in Fluids

Lecordier B, Trinite M (2003) Advanced PIV algorithms with image distortion validation and comparison using synthetic images of turbulent flow. In: Stanislas M, Westerweel J, Kompenhans J (eds) Particle image velocimetry: recent improvements. Proceedings of the EUROPIV 2 workshop, Springer, pp 115–132

Lecordier B, Trinité M (2006) Accuracy assessment of image interpolation schemes for PIV from real images of particle. In: Proc. 13th int. symp. on appl. of laser techn. to fluid mechanics, Lisbon, Portugal

Lecordier B, Westerweel J (2003) The EUROPIV Synthetic Image Generator (SIG). In: Stanislas M, Westerweel J, Kompenhans J (eds) Particle image velocimetry: recent improvements. EUROPIV 2 workshop, Springer

Miozzi M (2004) Particle image velocimetry using feature tracking and Delaunay Tessellation. In: Proceedings of the XII international symposium on application of laser technique to fluid mechanics, Lisbon

Pan B, Xie H, Guo Z, Hua T (2007) Full-field strain measurement using a two-dimensional Savitzky-Golay digital differentiator in digital image correlation. Opt Eng 46(3):1–10

Raffel M, Willert C, Wereley C, Kompenhans J (2007) Particle image velocimetry. A practical guide, 2nd edn. Springer, Heidelberg

Ruhnau P, Kohlberger T, Schnörr C, Nobach H (2005) Variational optical flow estimation for particle image velocimetry. Exp Fluids 38:21–32

Ruijters D, ter Haar Romeny BM, Suetens P (2008) Efficient GPU-based texture interpolation using uniform B-splines. J Graphics GPU Game Tools 13(4):61–69

Scarano F, Riethmuller M (2000) Advances in iterative multigrid PIV image processing. Exp Fluids 29:51–60

Schiwietz T, Westermann R (2004) GPU-PIV. In: Girod B, Magnor MA, Seidel HP (eds) Proceedings of the vision, modeling, and visualization conference, Aka GmbH, Stanford, CA, pp 151–158

Stanislas M, Okamoto K, Kähler CJ, Westerweel J (2005) Main results of the second international PIV challenge. Exp Fluids 39:170–191

Stanislas M, Okamoto K, Käler CJ, Westerweel J, Scarano F (2008) Main results of the third international PIV challenge. Exp Fluids 45:27–71

Venugopal V, Patterson C, Shinpaugh K (2009) Accelerating particle image velocimetry using hybrid architectures. In: Proceedings of symposium on application accelerators in high performance computing (SAAHPC'09), Urbana, Illinois

Westerweel J (1993) Digital particle image velocimetry—theory and application. PhD thesis, University Press, Delft

Westerweel J (2000a) Effect of sensor geometry on the performance of PIV interrogation. Laser techniques applied to fluid mechanics. Springer, Berlin, pp 37–55

Westerweel J (2000b) Theoretical analysis of the measurement precision in particle image velocimetry. Exp Fluids 29:3–12

Westerweel J, Dabiri D, Gharib M (1997) The effect of a discrete window offset on the accuracy of cross-correlation analysis of digital PIV recordings. Exp Fluids 23(1):20–28

Yamamoto Y, Uemura T (2009) Robust particle image velocimetry using gradient method with upstream difference and downstream difference. Exp Fluids 46(4):659–670

Yu H, Leeser M, Tadmor G, Siegel S (2006) Real-time particle image velocimetry for feedback loops using FPGA implementation. J Aerosp Comput Inf Commun 3(2):52–62

Zhao W, Sawhney H (2002) Is super-resolution with optical flow possible? In: ECCV02, Proceedings of the European conference on computer vision, pp 153–162