Abstract This paper presents a high-speed implementation of an optical flow algorithm which computes in real-time planar velocity fields in an experimental flow. Real-time computations of the flow velocity field allow the experimentalist to have instantaneous access to quantitative features of the flow. This can be very useful in many situations: fast evaluation of the performances and characteristics of a new setup, design optimization, easier and faster parametric studies, etc. It can also be used as a visual sensor for an input in closed-loop flow control experiments where fast estimation of the state of the flow is needed. The algorithm is implemented on a graphics processor unit. The accuracy of the computation is demonstrated. Computation speed and scalability of the processing are highlighted along with guidelines for further improvements. The system architecture is flexible, scalable and can be adapted on the fly in order to process higher resolutions or achieve higher precision. The setup is applied on a backward-facing step flow in a hydrodynamic channel. For validation purposes, classical particle image velocimetry (PIV) is used to compare with instantaneous optical flow measurements. The important flow characteristics like the dynamics of the recirculation bubble, computed in real time for the first time, are well recovered. The accuracy of real-time optical flow measurements is comparable to off-line PIV computations.

Keywords GPU · Real-time · Flow control · Measurements

1 Introduction

Optical measurements of 2D-velocity fields in fluid mechanics have been widely used in industrial and academics laboratories for more than a decade. They allow for the thorough investigation of flow physics through non-intrusive means and are an invaluable tool for understanding the dynamics of complex flows. The classical measurement technique is the standard 2D2C particle image velocimetry (PIV) which gives access to the 2-components (2C) of the velocity field in a 2D plane (Adrian 2005). It consists in illuminating the seeded flow with a plane laser sheet (typically generated by a pulsed YaG laser) and acquiring two images of the illuminated particles field at two successive time steps using 15-Hz double-frame cameras or fast cameras for time-resolved (1 kHz) measurements. Usually, a few hundreds pairs of images are acquired. In these standard PIV setups, data are transferred or stored on the computer and post-processed off-line because the computations to obtain a well-defined velocity field with a good spatial resolution (typically a $16 \times 16$ cross-correlation window) are time-consuming.

The development of reliable, flexible, accurate and low-cost systems capable of computing flow velocity fields in real time would be a great step forward for the fluid mechanics community. In addition to saving a lot of
time and resources, it would allow academics and industrial researchers to visualize the flow velocity field directly and make adjustments to their experiments on the fly. Accurately targeted measurement campaigns would become feasible even for flows exhibiting high-frequency behaviors, like flows downstream a bluff body (Pastoor et al. 2008; Joseph et al. 2012), a cylinder (Roshko 1961; or a wing Osbron et al. 2004).

Furthermore, such systems would open new perspectives for closed-loop flow control experiments based on visual information instead of wall-pressure or skin friction measurements. For instance, Henning and King (2007) used 4 × 15 microphones in parallel rows to measure pressure fluctuations downstream of a step. Using quantitative visual information would be equivalent to mapping the flow with as many captors as the image size divided by the spatial resolution of the 2D velocity field. Visual servoing in flow control has already been suggested and successfully implemented in numerical simulations (Fomena and Collewet 2011). One can find experimental demonstrations on improvement of the aerodynamic properties of micro-air vehicles (C. Willert 2010), or control of the flow behind a flap (Roberts 2012). Achieving increased performances would allow for additional means of control, such as vortex tracking or slope-seeking (Henning and King 2005). Several approaches have been suggested to achieve real-time PIV. For instance, a bare bones PIV algorithm has been implemented by (C. Willert 2010) on a single processor, obtaining engaging performances, while Roberts (2012) has implemented a basic PIV algorithm on a GPU. However, these approaches led to velocity fields from small images at relatively low frame rates (less than 20 fps). Direct cross-correlation PIV, and particle tracking velocimetry (PTV) algorithms have been programmed into field programmable gate arrays (FPGA) (Lelong et al. 2003; H. Yu and Siegel 2005; Kreizer et al. 2010). However, a specific hardware description language (HDL) is required to successfully operate them, which is a strong limitation. The spectacular increase in computing power of GPUs (the peak GFLOPS performance roughly doubles every year) allows for an alternative means of achieving real-time processing. Indeed, the processing power of graphics cards has risen at a rate superior to that of central processing units (CPU, doubles every two years). Until recently, it was difficult for the layman to access that power for something other than specific applications. With the introduction of GPU extensions for mainstream computing languages (C/C++, Fortran, Python, Matlab), implementing GPU code in a flexible manner has become accessible to a broader population.

A comprehensive overview of the algorithms used to compute flow velocity fields can be found in Heitz et al. (2010). In the present experimental study, a dense optical flow algorithm developed by Besnerais and Champagnat (2005) was used. Its characteristics and performances in comparison to PIV algorithms are comprehensively detailed in Champagnat et al. (2011) and Champagnat et al. (2009). This algorithm is notable for its performance. Performance of the algorithm increases hand in hand with GPU computing power. While optical flow algorithms have been used before to compute flow velocity fields they have never, to our knowledge, been used in a real-time setup.

Furthermore, a traditional PIV setup can be cheaply upgraded to a real-time PIV setup. To demonstrate the efficiency and quality of real-time velocity computations, it has been tested on a backward-facing step flow. Boundary layer separation and reattachment occur in many natural and industrial systems, such as diffusers, combustors or external aerodynamics of ground or air vehicles. The backward-facing step is the simplest geometry to study a separated flow. Though the geometry is simple, the complexity of separated flows is recovered as shown in Fig. 1. In this case, the separation is imposed by a sharp edge, allowing for the separation–reattachment process to be examined by itself. A dominant, global feature of the flow is the creation of a large recirculation bubble downstream the step edge, as shown in Fig. 1. This flow has been extensively studied through experimental and numerical investigations; see Armaly et al. (1983); Chun and Sung (1996); Hung et al. (1997); Beaudoin et al. (2004); Aider et al. (2007). As the objective of the present paper is exclusively the experimental demonstration of high-speed, efficient and reliable real-time velocity measurements, the BFS flow characteristics will not be discussed thoroughly, but solely used as a valuable benchmark for this experimental technique.

2 Experimental setup

2.1 Water tunnel

Experiments were carried out in a hydrodynamic channel in which the flow is driven by gravity. The walls are made of Altuglas for easy optical access from any direction. The flow is stabilized by divergent and convergent sections separated by honeycombs. The test section is 80 cm long with a rectangular cross
section 15 cm wide and 10 cm high. The mean free stream velocity \( U_\infty \) ranges between 1.38 to 22 cm.s\(^{-1}\). The quality of the main stream can be quantified in terms of flow uniformity and turbulence intensity. The standard deviation \( \sigma \) is computed for the highest free stream velocity featured in our experimental setup. We obtain \( \sigma = 0.059 \) cm.s\(^{-1}\) which corresponds to turbulence levels \( \frac{\sigma}{U_\infty} = 0.0023 \).

2.2 Optical flow measurement setup

The flow is seeded with 20 \( \mu \)m polyamide seeding particles. The vertical middle plane of the test section is illuminated from above (Fig. 2) by a laser sheet created by a 2W continuous CW laser operating at a wavelength \( \lambda = 532 \) nm.

The pictures of the illuminated particles are recorded using a relatively low-cost (compared to double-fame or high-speed cameras traditionally used for PIV), Basler acA 2000-340km 8bit CMOS camera, with a maximum bandwidth of 680 Mb/s. Its resolution is 2,048 \( \times \) 1088 pixels. The maximum frame rate for full-frame acquisition is \( F_{\text{acq}} = 340 \) Hz. The camera is controlled by a camera-link NI PCIe 1433 frame grabber allowing for real-time acquisition and processing. It should be noted that CPU performance is irrelevant with regards to the performance of the optical flow algorithm which runs entirely on the GPU. In our setup, a NVIDIA GeForce 580 GTX GPU card, with 520 processing cores clocked at 800 MHz, has been used. A complete description of this GPU’s architecture can be found in nVidia (2010). The data flow for the acquisition apparatus is detailed in Fig. 2. The images can either be written to a solid-state drive or computed in real time on the GPU. Usually no data is written during visualization of velocity fields to improve performance and frequency rate of the computation. The optical flow algorithm and camera acquisition software are integrated into a single interface using LabView. It is important to emphasize the only requirement to upgrade a classic PIV setup featuring a camera streaming images to an acquisition computer, to a setup capable of real-time flow velocity computations is adding a graphics card to the acquisition computer. Therefore, this can be done cheaply and with minimal effort.

2.3 Backward-facing step geometry

The backward-facing step geometry is shown in Fig. 2. A specific leading-edge profile is used to smoothly start the boundary layer which then grows downstream along the flat plate, before reaching the edge of the BFS. The boundary layer has a shape factor \( H \approx 2 \). Step height \( h \) is 15mm allowing for a range of Reynolds numbers \( 0 < Re_h = \frac{U_1 h}{v} < 3000 \), \( v \) being the kinematic viscosity.

2.4 Optical flow algorithm

Optical flow is related to the domain of image motion or optical flow estimation in computer vision. This particular algorithm called FOLKI was written in C++/CUDA. It was developed, implemented and rigorously validated by Champagnat et al. (2011) at ONERA. To achieve optimal performances further improvements were made by improving memory transfers and enhancing kernel concurrency. A guide on CUDA programming is available in nVidia (2007). This algorithm was used by Davoust et al. (2012), Sartor et al. (2012), and Rabinovitch et al. (2012). It is a local iterative gradient-based cross-correlation
optimization algorithm which yields dense velocity fields, i.e., one vector per pixel. It belongs to the Lucas–Kanade family of optical flow algorithms (Lucas 1984). It should be noted that the dense nature of the output is intrinsically tied to the nature of the algorithm. The spatial resolution, however, is tied to the window size, like any other window-based PIV technique. However, the dense output is advantageous since it allows the sampling of the vector field very close to obstacles, yielding good results near walls, as shown in Champagnat et al. (2011). Computing dense fields allows for a highly parallel algorithm which can take full advantage of the GPU architecture. The interrogation window radius $r = 10$ pixels was chosen, following the guidelines given by Champagnat et al. (2011). It should be noted that similar performances would be achieved using other programming languages, such as OpenCL in conjunction with any GPU, though the algorithm would need to be tweaked for the specific GPU architecture.

The principle of the featured optical flow algorithm is as follows. The original images are reduced in size by a factor of four iteratively until intensity displacement in the reduced image is close to zero. This gives a pyramid of images as illustrated in Fig. 3. The displacements are computed in the top image with an initial guess of zero displacement using an iterative Gauss–Newton scheme to minimize a sum of squared difference criterion. This displacement is then used as an initial estimate for the same scheme in the next pair of images in the pyramid. The process is repeated until the base of the pyramid which corresponds to the initial image, thus giving the final displacements field.

The optical setup is tuned for the displacement of the particles to be small enough for the optical flow algorithm to converge. Thus there are two inputs to the algorithm, besides image size, that have a major impact on performance: the number of levels in the pyramid $n_{lev}$, and the number of iterations per level $n_{iter}$ required to achieve convergence of the velocity field. Computing speed is a function of these two integers. If $(\delta x)_{\text{max}}$ is the maximum displacement, as a general rule $n_{lev}$ must verify Eq. 1 from Champagnat et al. (2011):

$$ (\delta x)_{\text{max}} / 2^{n_{lev} - 1} < 3 \text{pixels} $$

One can see that choosing the time step between images defines the value of $n_{lev}$. One must then choose $n_{iter}$. A low value will give higher performances with slightly lower result quality. When working in real time a low value (1 or 2) of $n_{iter}$ is recommended. However, for off-line computations the value should be raised to ensure full convergence. Performances should still be greater than with commercial PIV software.

While a number of pre and post-processing options are usually used to enhance the computed velocity fields, these operations have a computational cost. Therefore, a balance must be found between obtaining usable data and processing speed.
Raw images are pre-processed using a standard local equalization algorithm. This step is implemented on the GPU for increased performance. We have found this step to be mandatory for experimental images processing. Without, the computation does not yield usable data.

Original image intensity is normalized following Eq. 2:

\[
\forall (x,y) \in I, \tilde{I}(x,y) = \frac{I(x,y) - \overline{I}(x,y)}{\sqrt{\overline{I^2}(x,y) - \overline{I}(x,y)}}
\]

where $\overline{I}$ is a local mean for a given radius. We choose a radius of 5 pixels. This zero normalized sum of square differences (ZNSSD) is common for PIV pre-filtering. The aim is to eliminate the influence of illumination inhomogeneities. For each pixel the mean intensity value is subtracted (zero mean) and divided by the local intensity standard deviation (normalized).

The nature of the algorithm is such that computed velocity fields are naturally smooth. Thus there is no post processing required to cull spurious vectors.

For our setup, only a fraction of the camera resolution is used (the region of interest (ROI) = 1792 × 384). It is enough to capture the whole recirculation bubble downstream the BFS, while ensuring good computing performances. Reducing the time step $\Delta t$ between two pictures acquisitions allows lower values of $n_{\text{lev}}$ and higher performances. $n_{\text{lev}}$ can be lowered to 0 with a small enough displacement. Decreasing $n_{\text{lev}}$ shifts the burden of performance to the camera. $n_{\text{iter}}$ should be raised until the computed velocity field does not vary, with $n_{\text{iter}} \geq 1$. $n_{\text{iter}} \geq 10$ is seldom needed. With current hardware it is difficult to achieve satisfactory performances with a high number of iterations ($n_{\text{iter}} > 4$). $n_{\text{iter}}$ can be brought down as low as 1 and still yield usable quantitative information on the flow, with a significant improvement in computing times.

**Fig. 3** Sketch of the computation pyramid. The resolution of the original picture is divided by two in both directions $n$ times leading to coarser and coarser images with smaller and smaller displacements. The displacements are first evaluated on the coarser level which can then be used as first guess for the next level. We illustrate the case of a pyramid with $n = 2$ level.
Concerning latency, depending on GPU performances and camera acquisition frequency, different computing schemes are implemented for optimal performances as shown in Fig. 4.

The first scheme is used when computation is fast enough to keep up with the camera. This is the fastest scheme by far since each field computation requires only one image to be processed. The second is used when the first cannot and pre-processing time is lower than camera exposure. Finally, when preprocessing takes too long, preprocessing on the second image starts while preprocessing on the first image is finishing. Latency varies depending on the scheme, but is upper bounded by exposure time plus the time required to preprocess one image and compute the corresponding field. Post-processing can also be hidden during copy from the camera to the GPU, a period during which the GPU is idle. Post-processing here refers to the computation of integral quantities from the data.

2.5 PIV computations

To validate the optical flow measurements standard PIV algorithms were used off-line. The Davis software from LaVision was used, using a PIV multi-pass cross-correlation algorithm with a final 16×16 pixel interrogation window with 50% overlap, thus leading to PIV fields with a 8×8 pixel grid resolution.

3 Results

3.1 Real-time computation of instantaneous 2D velocity fields

Figure 5a shows the instantaneous velocity amplitude for $Re_h = 2500$ obtained using PIV. Figure 5b shows the velocity amplitude computed from the same pair of images using the optical flow algorithm. One can see that the instantaneous shear layer and recirculation bubble are well captured and very similar in both cases. To evaluate more quantitatively the difference between the two instantaneous velocity fields, the non-dimensional difference between the two fields is computed and shown on Fig. 5c. The differences are small over the whole velocity field, apart from small spots where difference are higher. This can be explained by small local shift of spots where velocity gradients are high. The differences between the two velocity fields are also mainly due to poorer results for the PIV algorithm near the edges of the acquisition window. One can also notice a poorly resolved region in the PIV field right after the step. This illustrates how the optical flow output can sometimes be superior to PIV even in the free stream region.

Fig. 4 Different computing schemes
3.2 Comparison of the real-time optical flow measurements with off-line PIV computations

The accuracy of the algorithm has been demonstrated off-line for numerical and experimental data by Champagnat et al. (2011). In this section, we will focus on the computation of an integral scalar value derived in real time from the instantaneous velocity fields. The objective is twofold: illustrate the real-time computation of a global quantity extracted from instantaneous velocity fields in real time and evaluate the accuracy of the optical flow computations compared to standard PIV computations.

There are a number of pertinent integral values which can be used to characterize the separated flow. In this experiment, we choose to compute a measure of the recirculation area in the instantaneous 2D velocity field. It is straightforward and quick to compute while remaining a good way of evaluating the state of the flow. Equation 3 describes how recirculation area is defined:

$$A_r = \int H(-v)(x, y)\,dx\,dy,$$

where $H$ is the Heaviside function and $v$ is longitudinal velocity. In the following $A_r$ will be normalized by $h^2$. It should be noted this is one way of defining recirculation area. Another definition is the region bounded by the wall and the streamline connecting the separation point at the step edge and the reattachment point, the upper half of this area has positive longitudinal velocity. Thus the former definition for recirculation area would be about half of the latter.

Figure 6 shows the recirculation area (in black) for an instantaneous velocity field. Such computations are carried out for both optical flow and PIV velocity fields. The recirculation bubble area can be correlated
to the reattachment length $L_R$ usually used to characterize the BFS flow (Armaly et al. 1983; Aider et al. 2007). Such an integral value could, for example, be used as an input in a feedback loop.

Figure 7 shows time series of instantaneous $A_r$ computed with optical flow and with PIV for a time-resolved series computed off-line. Data featured in Fig. 7 are for $Re_h = 2500$ and with an acquisition rate of 60 image pairs per second. One can see that there is a good agreement between the two time-series. Differences can be explained by the fact some images do not contain enough particles in the recirculation region, thus both algorithm converge to slightly different values in this region; furthermore, window interrogation sizes are different (10 for optical flow, 16 for PIV) and overlap is also different (90% for optical flow, 50% for PIV). Because of the dense nature of the optical flow output, more information is available near the walls (Champagnat et al. 2011). The computed recirculation bubble is more clearly defined and is subject to greater variations as shown in Fig. 7.

Figure 8a shows a comparison of the surface of the mean recirculation bubble as a function of Reynolds number computed by PIV or optical flow. Figure 8b shows the relative difference between results obtained with both algorithms. Agreement is good with a relative difference always lower than 3%. It shows that the optical flow computations is robust over a wide range of Reynolds numbers corresponding to complex instantaneous flows.

3.3 Optimizing the computation frequency

This setup allows for accurate computations of the 2D velocity fields in real time. If the aim of the user is real-time visualizations and/or quick computations of flow properties for a feedback loop, constraints on the algorithm can be relaxed. Indeed, some compromise can be found if increased computations speed is called for. If the aim is the computation of accurate velocity fields, constraints should be increased to achieve maximum accuracy, while still retaining fast computing times in real time.

Table 1 shows how one can drastically improve computation speed, by lowering the number of levels and iterations while still retaining a meaningful integral information from the flow. $n_{iter}$ is the most important parameter when it comes to performance, with $n_{iter}$ a distant second. This is understandable as higher tiered levels of the pyramid have much smaller images sizes causing little increase in processing costs.

A video is linked to this article showing flow images and real-time computed fields as well as $A_r$ history. Higher frame rate can be achieved by shortening the time step between two images in order to lower the maximum displacement and thus allowing $n_{lev} \to 0$. It can lead to a maximum of 350 fps at this resolution. Higher sampling frequency can be achieved by reducing the field of view, for example, by focusing on an area of interest or using masks to avoid computations over obstacles or side walls.

The aforementioned performances are achieved with a given hardware. The scalability of the algorithm ensures greater performance with better GPUs. Moreover, since the computation of a velocity field is
independent of all other computations, adding additional graphic cards to the setup would allow for a proportional increase in computation speed. The only limit to the achievable frame rate is the acquisition rate of the camera image quality. For very high acquisition frequencies, a more powerful CW laser, a pulsed YaG laser and/or a more sensitive and faster camera are required.

4 Conclusion and perspectives

We have shown how a simple and relatively low-cost setup can be configured to achieve high-speed real-time computations of a flow velocity field. The key feature of the setup is the use of an optical flow algorithm which takes advantage of the massively parallel processing capabilities of GPUs. It is now possible to compute in real time any local or global relevant quantity from this velocity field. It is now easy to evaluate the state of a complex flow in real time, which a major step forward for experimental fluid mechanics community. The relevant flow characteristics can be computed and stored without keeping the raw data (images time series). This can lead to major savings in both time and data storage facilities. This work is useful to experimentalists who wish to quickly analyze flow properties, it can also be useful to those who wish to use high-frequency flow data to implement closed-loop control in flow control experiments. We have demonstrated the accuracy of the method by comparing our results with results obtained by the more widely used PIV approach, computed off-line. Finally, ways of improving computing speed and reaching higher frame rates have also been discussed.

Acknowledgments The authors gratefully acknowledge the financial support of the DGA, as well as Aurelien Plyer (ONERA) for all his helpful advices.
References

Adrian RJ (2005) Twenty years of particle image velocimetry. Exp Fluid 39(2):159–169
Aider JL, Danet A, Lesieur M (2007) Large-eddy simulation applied to study the influence of upstream conditions on the time-dependent and averaged characteristics of a backward-facing step flow. J Turbul 8:N51
Armaly BF, Durst F, Pereira JCF, Schonung B (1983) Experimental and theoretical investigation of backward-facing step flow. J Fluid Mech 127:473–496
Beaudoin JF, Cadot O, Aider JL, Westfreid JE (2004) Three-dimensional stationary flow over a backwards-facing step. Eur J Mech 38:147–155
Bruce D (1984) Lucas Carnegie Mellon University, Generalized image matching by the method of differences. Phd
Champagnat F, Pfyer A, Le Besnerais G, Leclaire B, Le Sant Y (2009) How to calculate dense piv vector fields at video rate
Armaly BF, Durst F, Pereira JCF, Schonung B (1983) Experimental and theoretical investigation of backward-facing step flow. J Fluid Mech 127:473–496
Beaudoin JF, Cadot O, Aider JL, Westfreid JE (2004) Three-dimensional stationary flow over a backwards-facing step. Eur J Mech 38:147–155
Champagnat F, Plyer A, Le Besnerais G, Leclaire B, Davoust S, Le Sant Y (2011) Fast and accurate piv computation using highly parallel iterative correlation maximization. Exp Fluids 50:1169–1182 (ISSN 0723–4864)
Chun K B, Sung H J (1996) Control of turbulent separated flow over a backward-facing step by local forcing. Exp Fluids 21:417–426 (ISSN 0723–4864)
Davoust S, Jacquin L, Leclaire B (2012) Dynamics of \( m = 0 \) and \( m = 1 \) modes and of streamwise vortices in a turbulent axisymmetric mixing layer. J Fluid Mech 709:408–444
Fomena RT, Collewet C (2011) Fluid flow control : a vision-based approach. Intl J Flow Control 3:133–169
Gharib M, Willert C, Munson M (2010) Real-time particle image velocimetry for closed-loop flow control applications. vol. 1, pages 1–1. 5th Intl Symposium on Applications of laser techniques to fluid mechanics, 05–08 Jul 2010, Lisb Port
Heitz D, Me¨min E, Schno¨rr C (2010) Variational fluid flow measurements from image sequences : Synopsis and perspectives. Exp Fluids 48:369–393
Henning L, King R (2005) Drag reduction by closed-loop control of a separated flow over a bluff body with a blunt trailing edge. 44th Institute of Electrical and Electronics Engineers (IEEE) Conference on Decision and control and European Conference (ECC), IEEE, Piscataway, NJ 1:494–499
Heeling Lars, King Rudibert (2007) Robust multivariable closed-loop control of a turbulent backward-facing step flow. J Aircr 44:201–208
Hung L, Parviz M, John K (1997) Direct numerical simulation of turbulent flow over a backward-facing step. J Fluid Mech 330:349–374
Joseph P, Amandolese X, Aider JL (2012) Drag reduction on the 25 degrees slant angle ahmed reference body using pulsed jets. Exp Fluid 52:1169–1185
Kreizer M, Ratner D, Liberzon A (2010) Real-time image processing for particle tracking velocimetry. Exp Fluid 48:105–110
Lelong L, Motyl G, Dubois J, Aubert G, Jacquet G (2003) Image processing in fluid mechanics by cmos image sensor. In: Proceedings of PSFVIP-4
Le Besnerais G, Champagnat F (2005) Dense optical flow by iterative local window registration. ICIP 1:137–140
nVidia (2007) Cuda c programming guide
nVidia (2010) White paper, fermi architecture
Osbron R, Kota S, Hetrick J, Geister D, Tilmann C, Joo J (2004) Active flow control using high-frequency compliant structures. J Aircr 41:603–609
Pastoor M, Henning L, Noack BR, King R, Tadmor G (2008) Feedback shear layer control for bluff body drag reduction. J Fluid Mech 608:161–196
Rabinovitch J, Brion V, Blanquart G (2012) Effect of a splitter plate on the dynamics of a vortex pair. Phys Fluid 24:74–110
Roberts J (2012) Control of underactuated fluid-body systems with real-time image velocimetry. PhD, MIT
Roshko A (1961) Experiments on the flow past a circular cylinder at very high reynolds number. J Fluid Mech 10:345–356
Sartor F, Losfeld G, Bur R (2012) Piv study on a shock-induced transition in subsonic flow. Exp Fluid 53:815–827
Tadmor G, Yu H, Leeser M, Siegel S (2005) Real-time particle image velocimetry for feedback loops using fpga implementation. 43rd AIAA Aerospace Sciences Meeting and Exhibit