*Chapter 4*

# RAPIDPIV: FULL FLOW-FIELD KHZ PIV FOR REAL-TIME DISPLAY AND CONTROL

**Preamble**

The primary goal of the work in this chapter was to allow PIV to become an easily accessible and interactive tool, while allowing researchers to simultaneously save time and hard-drive space during diagnostic and exploratory work, and at the same time providing results of sufficient quality to warrant direct publication of recorded results without re-processing of raw images with other tools. However, in the course of performing the work it was noticed that the software in-development had all of the capabilities required to perform as a general full-field sensor for fluid feedback control applications at high frequencies, unlocking capabilities that real-time PIV systems have historically not been able to achieve.

During the development of this work an entirely different method of achieving real-time PIV, HyperFlow, was discovered by the authors. It achieves nearly 1 million frames per second full-field processing rates on 1MP images. Since HyperFlow is not an expression of Scott Bollt's independent work, Sam Foxman being first author, it is not included in this thesis. However, between RapidPIV and HyperFlow, we posit that real-time PIV in 2D is essentially a solved problem, since no streaming camera currently manufactured can produce data as fast as it can be processed by HyperFlow, and the quality tradeoffs usually present in real-time PIV systems are largely gone in RapidPIV. A more in-depth analysis of the historical trends in RTPIV, GPU performance, camera capabilities, and data channel capacity is provided in appendix A.3.1.

*This chapter is adapted from: S. A. Bollt, S. H. Foxman, and M. Gharib, "RapidPIV: full flow-field kHz PIV for real-time display and control," Measurement Science and Technology, 36(10), 2025. doi:10.1088/1361-6501/ae124d. The article was published as open access, giving free license for re-use as a thesis chapter.*

## 4.1 Introduction

Particle Image Velocimetry (PIV) is a method by which the flow velocity field in a 2D or 3D domain may be measured by imaging tracer particles which passively

advect with the flow. It is almost peerless in its ability to extract the fluid's entire hydrodynamic state (for homogeneous, incompressible flows). In its most basic form PIV works by illuminating a plane of particles suspended in a fluid with a laser sheet. Images of the particles' motion are recorded with a camera, and from these images, the motion of the fluid is inferred.

With the introduction of digital methods, PIV became more repeatable and much less laborious because images could be transferred directly from digital cameras to digital computers where the images can be processed algorithmically Willert and Gharib, 1991. Digital PIV (DPIV) has since become ubiquitous in research as a flow diagnostic and a-posteriori measurement tool with a wide range of applications.

Real-time PIV (RTPIV) is an implementation of PIV which is capable of producing vector fields from measurements with sufficiently little delay that the velocity fields are representative of the current state of the fluid as it pertains to the time-scale of interest. RTPIV expands the applicability of PIV in a few key ways. Because the velocity fields are being produced as the flow evolves, it becomes possible to integrate RTPIV into control schemes both in research and industrial settings so long as the velocity field is updated substantially faster than the characteristic growth rate of the flow instabilities of interest. The flexibility and spatial resolution of PIV allows for sophisticated measurement and control schemes which are impractical or impossible with other kinds of sensors. The architecture and performance of RapidPIV makes it compatible with flow control, but the software itself does not support it. However, it does provide low-latency display of velocity fields as they are processed. That means it provides a visual display which is "real-time" to humans if one considers the timescale of interest to be the time over which our brain integrates visually detected motion.

RTPIV can also be used in situations where low latency is less important, because the throughput alone of RTPIV enables types of experiments or analysis which would simply be impossible otherwise. For instance, high throughput PIV processing with moderate latency (which RapidPIV is capable of) can enable expanded experimental campaigns either by reducing cycle time between experiments or increasing experiment runtime. Usually the processing time of PIV data exceeds the experiment time by orders of magnitude. For instance, recent experiments on the wakes of birds flying in formation used 148 recordings of 10 second 8MP video taken at 500 Hz Hao, 2025. From communication with the author, processing this

entire dataset takes 40 days of compute time [1], so only a subset of the recorded data could actually be used for analysis, and it is a difficult and time consuming procedure to experiment with different setups and processing parameters.

Various authors have contributed to the state of the art of RTPIV since its introduction in 1997 Arik and Carr, 1997. Indeed, a speedup exceeding an order of magnitude was achieved in the following two decades Champagnat, Plyer, Le Besnerais, Leclaire, Davoust, and Le Sant, 2011; Gautier and Aider, 2014,Pimienta and Aider, 2024; Wernet, 2019. Event-based camera methods have more recently come into favor and offer an alternative approach with impressive performance, but an entirely different set of trade-offs and performance measures Drazen, Lichtsteiner, Häfliger, Delbrück, and Jensen, 2011; Rusch and Rösgen, 2023. However, RTPIV has yet to see wide adoption even in research settings despite its theoretical and practical advantages. One of the major roadblocks to widespread adoption is the difficulty involved in setting up and implementing RTPIV. A RTPIV system requires delicately crafted code which, with exception of Willert, Munson, and Gharib, 2010b, is run on specialized hardware such as FPGAs, or more recently GPUs. Tight coupling to a video streaming camera is also required. For this reason, historically RTPIV systems have been built from the ground up by research groups around a specific camera and compute hardware. Furthermore, until recently few fluid systems of practical importance were within the range of proven RTPIV capabilities (as an exception to this rule, see Varon, Aider, Eulalie, Edwige, and Gilotte, 2019). Regardless, for existing systems the fluid needs to evolve slow enough that dozens of frames per second can be considered real-time. Most existing RTPIV systems also cannot handle large displacements, large gradients, or most of the other challenges which naturally appear in typical PIV applications because they are based upon single-pass correlation with minimal pre and post-processing.

We address these challenges, in part, with RapidPIV: a software which can stream, from camera or disc, and process image data at over 1kHz. The result is a vast reduction in processing time and storage needs, as well as real-time PIV display/visualization. RapidPIV offloads the initial stage of the PIV workload to Nvidia's purpose-built optical flow hardware accelerator (NVOFA) Zhang, Lu, and Hu, 2019. The NVOFA is a specialized circuit in all Turing, Ampere, and Ada generation Nvidia GPUs (2018-2024). The sole purpose and capability of the Ada generation NVOFA is performing the memory-efficient Semi-Global Match-

[1] The author used a Windows 11 desktop with an Intel i9-11900K CPU and an NVIDIA 4070 Ti GPU. The PIV software was Davis 10.2 from LaVision.

ing (eSGM) algorithm as quickly and power efficiently as possible Hirschmuller, 2008. Because it was built to perform a single task, many electrical and logic optimizations are possible which are not possible on a general purpose computer. Camera communication, data handling, GUI, pre-processing, and post-processing code were developed in-house. The program is designed to be easy to use and as hardware-agnostic as possible (beyond needing an Nvidia GPU). The software works in Windows and Linux and uses GenICam, the streaming camera interface standard run by the European Machine Vision Association, so it is compatible with cameras that use GigE Vision, USB3 Vision, CoaXPress, Camera Link HS, Camera Link, etc. The GUI allows users to view various real-time overlays at screen refresh rate such as velocity and vorticity (the program processes the data at the camera frame rate regardless of the screen refresh rate). Offline processing of image sequences with simultaneous display are also supported.

## 4.2 RapidPIV

RapidPIV is an app for Windows and Linux, with a CUDA-accelerated image processing pipeline centered around the NVOFA. RapidPIV is able to stream images from GenICam-compatible cameras, or load pre-existing particle image files. It calculates and displays processed velocity fields in real-time (up to the limit shown in figure 4.5 and the screen refresh rate, respectively), and saves results to `.mat` files suitable for further analysis in Python or MATLAB. The GUI display is shown in the appendix, figure A.11. In order to robustly generate precise and sub-pixel accurate velocity fields, RapidPIV pre-processes particle images before executing the NVOFA, and refines flow vectors afterwards with classical PIV techniques. The processing pipeline is shown in figure 4.1. We now cover these stages in more detail.

### 4.2.1 NVOFA Algorithmic Overview

The NVOFA is a proprietary hardware accelerator, designed by Nvidia, which exists in Turing, Ampere, and Ada generation Nvidia GPUs. The NVOFA calculates optical flow between two input images given an initial displacement vector estimate. An exact account of how it works was not made public by Nvidia. However, a patent filed by the company, Zhang, Lu, and Hu, 2019, reveals enough about how Ada generation chips perform optical flow that it is possible to make qualitative statements about the expected performance, and understand its strengths and weaknesses. To a large degree, the algorithm implemented in NVOFA accelerates appears to be built on the one set out by Hirschmüller, Buder, and Ernst, 2012. As will be seen,

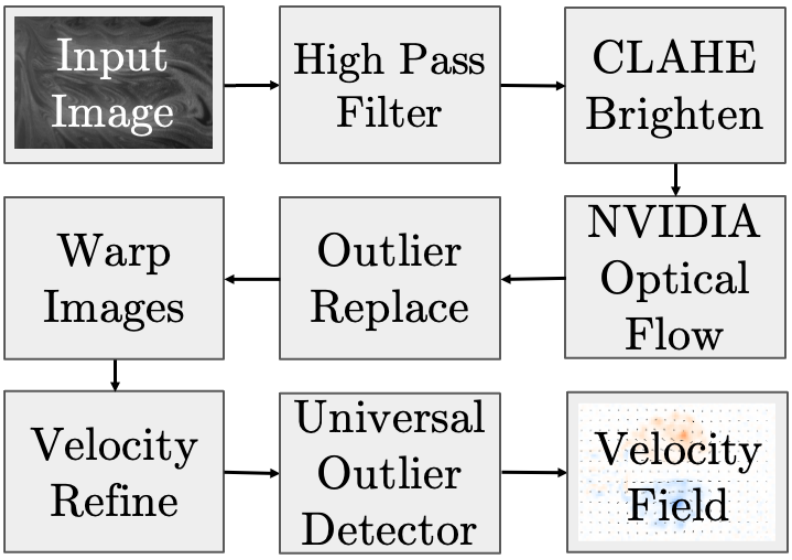


Figure 4.1: CUDA Processing Pipeline. RapidPIV filters and brightens images, calculates an initial velocity estimate with the Nvidia Optical Flow Accelerator, and refines the velocity field with classical PIV techniques.

the NVOFA on Ada-generation chips provide a robust estimate of displacement which is nonetheless incapable of the precision and accuracy demanded by PIV; the eSGM algorithm was designed to be fast while being sufficiently robust to handle the challenges of "natural" images (i.e., images one might take of their environment in daylight or room lighting). These qualities make NVOFA a good candidate for producing a rough first guess at the displacement field, even when the flow is complex or improperly seeded, but a bad candidate for an all-in-one PIV solution.

The NVOFA accepts two images as input, as well as a displacement field guess which we set to be the previous vector field. The two images are processed to form a Gaussian pyramid so as to represent the image at multiple scales (see Champagnat, Plyer, Le Besnerais, Leclaire, Davoust, and Le Sant, 2011 for more details on this technique and why Gaussian pyramids are useful in optical flow and PIV alike). All further steps of the algorithm are repeated at each level of the pyramid.

The first processing step is the census transform. The census transform, introduced by Zabih and Woodfill, 1994, compares the intensity of each pixel with its nearest neighboring pixels in a square grid, and outputs the Boolean values of these comparisons concatenated as a binary string. The census transform is useful for pulling out features to track, especially in regions of natural images with near-uniformity (like the texture on a white wall). For PIV, all of the information is contained in the particles, so the census transform is more aggressive than necessary, although

the transform's invariance with respect to local illumination makes it useful for preventing issues resulting from non-uniform lighting. The transform also loses detail about the shape of the particle distribution, which is normally used for sub-pixel accuracy in PIV. Therefore, we expect peak locking from NVOFA due to this census transform step.

After the census transform is performed, a 4D cost volume, $C(i, j, \underline{D}(i, j))$, is constructed piece-by-piece as needed. At each evaluation location, $(i, j)$, the cost volume assigns a matching penalty between a patch of the first image and a patch of the second, for different discrete displacements, $\underline{D}(i, j) \in \mathbb{Z}^2$, between the patches. This is almost identical to how the correlation plane in cross-correlation PIV assigns a value to the agreement between a patch in the first image and a patch in the second for different displacements. The only difference is that the metric of discrepancy for the cost volume is the Hamming distance of the census transform whereas in cross-correlation PIV it is the correlation. A value is assigned for a set number of horizontal and vertical patch displacements, $\underline{D}$, for each evaluation location in the image, $(i, j)$ (this makes it a 4D volume). Note that the stride for $(i, j)$ need not be 1 pixel, and in fact we choose to make it 4 pixels so that there are 16 times fewer evaluation locations than pixels. It is at this stage that the guess displacement field is used: costs are only computed for $\underline{D}$ sufficiently close to the displacement guess.

In order to find a $\underline{D}(i, j)$ which is representative of the motion that occurred between two images, Hirschmuller, 2008 defines the *global* energy function to minimize:

$$E(\underline{D}(i, j)) = \sum_{i,j} \left( C(i, j, \underline{D}(i, j)) + \sum_{\mathcal{S}_1(i,j)} P_1 + \sum_{\mathcal{S}_2(i,j)} P_2 \right). \tag{4.1}$$

In equation 4.1 the set $\mathcal{S}_1(i, j)$ includes all possible points and displacements so long as they are sufficiently near $(i, j)$ and the displacement is not the same as the one $\underline{D}(i, j)$, but also not different than it by more than some constant threshold (typically 1 pixel). The set $\mathcal{S}_2(i, j)$ is also restricted to points sufficiently near $(i, j)$, but its members have a displacement that is different from it at $(i, j)$ by more than the aforementioned threshold. Essentially, in addition to the cost volume penalty, at each evaluation point $(i, j)$ the penalty $P_1$ is applied for every evaluation point near $(i, j)$ for which $\underline{D}$ is different to its value at $(i, j)$ by some small amount. The larger penalty $P_2$ is applied when the displacements are different by a larger amount, and no penalty is incurred for neighboring displacements being the same. The use of constant penalties prevents discontinuities from being smoothed out

while also allowing for preservation of smooth gradients. This is a useful feature in handling natural images where rigid body motion in the foreground will produce discontinuous velocities with respect to the background, but it is rarely necessary in fluid mechanics outside of handling shocks or under-resolved boundary layers. The downside of this method is the displacement field tends to snap to neighboring displacements, not just to integer displacements.

Minimizing 4.1 is computationally intractable, so instead Hirschmuller, 2008 proposed solving a *semi-global* approximation to 4.1, which was yet again approximated to reduce memory requirements in eSGM Hirschmüller, Buder, and Ernst, 2012 (this later approach is the one Nvidia adapted for the Ada generation). The actual approach is quite involved, mostly for algorithmic and technical reasons rather than structural ones. For this reason, and because it is explained in the aforementioned references, we opt to only summarize the key changes in eSGM that make the results qualitatively different from minimizing equation 4.1 itself. Instead of minimizing 4.1 directly, the outer 2D sum over $(i, j)$ is replaced with a set of functions, $L_r(i, j, \underline{D}(i, j))$, which are each 1D sums. These 1D sums are pieced together by adding one evaluation location at a time to the function, starting from the top and bottom edges of the image, only evaluating the difference in displacement along 8 principle grid directions (up, down, left, right, and diagonals). In Hirschmüller, Buder, and Ernst, 2012 and Zhang, Lu, and Hu, 2019 this is done by performing a top-to-bottom raster scan and then a bottom-to-top scan followed by a final top-to-bottom scan. The first two scans are depicted in figure 4.2, while the third scan's purpose is to repeat the first scan in order to fill in missing information. These eight functions are summed to approximate the full 2D sum: $\tilde{E}(i, j, \underline{D}(i, j)) = \sum_r L_r(i, j, \underline{D}(i, j))$, although again for technical reasons this is done as part of the second and third raster scans.

The obvious benefit of attempting to minimize $\tilde{E}$ rather than $E$ is speed, but the drawback is that $\tilde{E}$ is not always a good approximation of $E$. This is especially true near the edges of the image. The approximation is worst of all at the top and bottom edges because that is where the construction of $L_r(i, j, \underline{D}(i, j))$ begins for some of the 8 directions. The consequence is that the NVOFA is most robust in the interior of images, less robust on the left and right edges, and the least robust on the top and bottom edges. It also can have trouble "re-starting" near regions where it finds incorrect matches since then $L_r(i, j, \underline{D}(i, j))$ from some of the directions which pass through this patch are built on bad information.

Once all of the above calculations are performed at each level of the Gaussian

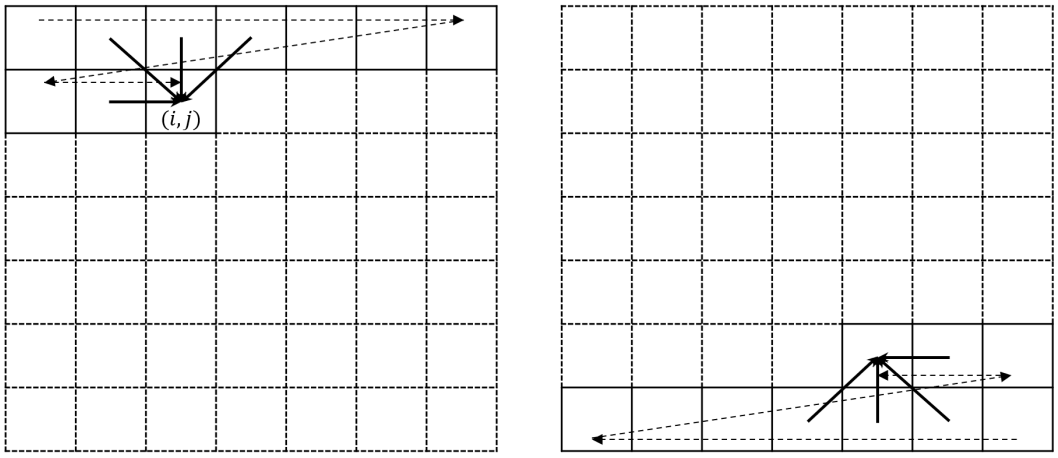


Figure 4.2: Depiction of how the first 4 $L_r$ are put together at the evaluation point $(i, j)$ in the top-to-bottom pass (left grid, pass 1). The second pass (right grid) constructs the other 4 $L_r$ from the bottom up. Dashed arrows show the raster scan ordering.

pyramid, the result is a displacement field at the base-level resolution. To achieve sub-pixel resolution, equi-angular interpolation is used by the NVOFA to estimate the true minimum of the function $\tilde{E}$ in the vicinity of $\underline{D}^*$, where $\underline{D}^*$ is the displacement which minimizes $\tilde{E}$. It is known that in certain cases equi-angular interpolation can reduce peak-locking Shimizu and Okutomi, 2002. Equi-angular interpolation estimates the first component of the sub-pixel displacement as,

$$\delta(i,j) = \frac{1}{2}\begin{cases} \dfrac{\tilde{E}\left(i,j,\underline{D}^*+\begin{bmatrix}0\\1\end{bmatrix}\right)-\tilde{E}\left(i,j,\underline{D}^*+\begin{bmatrix}0\\-1\end{bmatrix}\right)}{\tilde{E}\left(i,j,\underline{D}^*\right)-\tilde{E}\left(i,j,\underline{D}^*+\begin{bmatrix}0\\-1\end{bmatrix}\right)} & \Longleftrightarrow \tilde{E}\left(i,j,\underline{D}^*+\begin{bmatrix}0\\-1\end{bmatrix}\right) > \tilde{E}\left(i,j,\underline{D}^*+\begin{bmatrix}0\\1\end{bmatrix}\right) \\ \dfrac{\tilde{E}\left(i,j,\underline{D}^*+\begin{bmatrix}0\\1\end{bmatrix}\right)-\tilde{E}\left(i,j,\underline{D}^*+\begin{bmatrix}0\\-1\end{bmatrix}\right)}{\tilde{E}\left(i,j,\underline{D}^*\right)-\tilde{E}\left(i,j,\underline{D}^*+\begin{bmatrix}0\\1\end{bmatrix}\right)} & \Longleftrightarrow \text{else} \end{cases}. \tag{4.2}$$

The second component is found by varying the other displacement component. The displacement between images is then the sum of the discrete displacement and the sub-pixel correction: $\underline{d}'(i,j) = \underline{D}^*(i,j) + \delta(i,j)$.

Further details can be found in Zhang, Lu, and Hu, 2019, and Hirschmuller, 2008; Hirschmuller and Scharstein, 2009, and Hirschmüller, Buder, and Ernst, 2012. Clearly, the NVOFA algorithm makes compromises on sub-pixel accuracy in order to improve robustness and speed, with speed being the primary goal. In order to use the NVOFA for PIV, we will need to refine the displacement field so as to correct the bias and noise of the NVOFA's output. However, since the NVOFA was designed to work well even in the challenging environment presented by natural images, we should expect it to robustly get close to the correct displacement field.

### 4.2.2 RapidPIV Processing Details

As we saw above, the NVOFA quickly provides a dense displacement field which is usually correct to the nearest pixel, but not nearly as precise or accurate as we would like for PIV applications. The main purpose of our added pipeline is to improve on this. First, images are grabbed as soon as available from the camera or folder, and uploaded to the GPU memory. To pre-process images we optionally apply a spatial high-pass filter, by subtracting a Gaussian blurred version of each image of user-definable standard deviation $\sigma$, to remove static background. The Gaussian blur is achieved directly, as opposed to in the spectral domain, using a Gaussian kernel. Contrast-limited adaptive histogram equalization (CLAHE) Pizer, Amburn, Austin, Cromartie, Geselowitz, Greer, ter Haar Romeny, Zimmerman, and Zuiderveld, 1987 is also optionally used to brighten the image. Then, these images are sent to the NVOFA.

For each image captured, $n$, RapidPIV sends the pre-processed images $(I_n, I_{n-1})$ to the NVOFA where it evaluates the displacement between the image pairs in both ordinary and time-reversed order. This produces forward displacement vector field $\underline{d'}_n(i,j)$ and reverse displacement vector field $\overline{d'}_n(i,j)$, on the domain $(i,j) \in \mathcal{D}$ (in what follows sub-scripts will refer to the temporal index of a variable, while spatial location is denoted by the argument). In order to reduce latency the forward and time-reversed images are sent into the NVOFA at the same time by stacking them vertically. The reason we generate a forward and reverse vector field in spite of the substantial speed penalty this causes is that the NVOFA does not treat the two images equally, unlike cross-correlation. This both means that better results can be obtained by averaging the forward and reverse fields, and it is possible to detect incorrect vectors by comparing the forward and reverse fields: if both forward and reverse vectors are correct at a point in space they will agree with each-other, whereas if one or both vectors are an outlier the chance they agree is slim. The use of a

forward-reverse consistency check with SGM dates back to its origins Hirschmuller, 2008, and is used by Nvidia in their Frame-Rate Up-Conversion (NVOFA FRUC) capabilities in the Ada generation.

With a forward and reverse displacement field generated it is necessary to merge the displacement fields. The displacement merge step can be broken into two parts: first, an outlier region is identified. Next the forward and backwards vectors along with the previous vector field are combined in a manner which eliminates outliers, reduces random error, and minimizes lag. The goal of the merge step is to produce a displacement field which roughly approximates the true one. It is crucial to get within roughly 1 px of the true field to ensure the refinement step works accurately and reliably. However, because of the refinement step to come, an exact estimate at this stage is not necessary. After the merge step the displacement field is known to within 0.5 px, typically, at each point.

We identify a subset of the displacement field as outliers using a forward-reverse displacement field check: $O_n$ is the set of all points in $\mathcal{D}$ where the forward and backwards vectors disagree by more than a threshold, $\beta$, in the euclidean norm; $O_n = \left\{(i,j) \in \mathcal{D} : \left\| \underline{d'}_n(i,j) + \overline{d'}_n(i,j) \right\|_2 > \beta \right\}$.

With the outliers identified, the algorithm can treat the outlier and non-outlier regions separately to merge the displacement field. In the non-outlier region, $O_n^{\mathsf{c}}$, we average the forward and backwards estimates with equal weight to produce intermediate displacement field $\underline{\tilde{d}}_n(i,j)$; $\underline{\tilde{d}}_n(i,j) = \left(\underline{d'}_n(i,j) - \overline{d'}_n(i,j)\right)/2 \, \forall (i,j) \in O_n^{\mathsf{c}}$. Then, simple exponential smoothing, with controllable smoothing constant $\alpha_1$, is applied to take advantage of the small temporal correlation of the NVOFA's random errors in order to produce part of the validated displacement field, $\underline{d}_n(i,j)$; $\underline{d}_n(i,j) = \alpha_1 \underline{\tilde{d}}_n(i,j) + (1-\alpha_1)\underline{\tilde{d}}_{n-1}(i,j) \, \forall (i,j) \in O_n^{\mathsf{c}}$.

For points in the outlier region where the vector was valid at the previous time we may replace the current invalid estimate with the valid previous one: $\underline{d}_n(i,j) = \underline{d}_{n-1}(i,j) \forall (i,j) \in O_n \cap O_{n-1}^{\mathsf{c}}$. Finally, in the region where neither the current nor previous vectors are valid, we use a normalized kernel filter of width and height $2M+1$ to interpolate from valid neighbors,

$$\forall (i,j) \in O_n \cap O_{n-1}, \, \underline{d}_n(i,j) = \frac{\sum_{\mathcal{K}(i,j)} \underline{d}_n(k-i, l-j)(K_0(k,l))}{\sum_{\mathcal{K}(i,j)} K_0(k,l)}, \quad (4.3)$$

where the domain over which the sum is done is defined by the set $\mathcal{K}(i,j)$,

$$\mathcal{K}(i,j) \equiv (k,l) \in [-M,M] \times [-M,M] : (k-i, l-j) \in (\mathcal{O}_n \cap \mathcal{O}_{n-1})^{\mathsf{c}}. \quad (4.4)$$

In equation 4.3 the 0th order modified Bessel function of the second kind, $K_0$, was chosen as the kernel function because it heavily weights nearby points, and because it is the Green's function for the screened Laplace equation which has been shown to be effective at scalar field reconstruction in the presence of incomplete, noisy, and missing data Kazhdan and Hoppe, 2013; Bhat, Curless, Cohen, and Zitnick, 2008. Once the entire validated NVOFA displacement field is produced ($\underline{d}_n$), RapidPIV warps the pre-processed particle image pair. To do so, bilinear interpolation is used to upscale $\underline{d}_n$ as needed to form $\hat{\underline{d}}_n$, which assigns a displacement for each pixel in $I_n$. The images $I_n$ and $I_{n-1}$ are warped according to,

$$I_{n+}(a,b) = \mathcal{I}\left(I_n((a,b) + \hat{\underline{d}}_n(a,b)/2)\right), \quad (4.5)$$

$$I_{n-}(a,b) = \mathcal{I}\left(I_{n-1}((a,b) - \hat{\underline{d}}_n(a,b)/2)\right), \quad (4.6)$$

in order to obtain two separate estimates of what the particle images would have looked like had an image been taken halfway between image $n$ and image $n-1$. The function $\mathcal{I}$ performs bilinear interpolation to form an image by placing the displaced intensities onto integer locations. We use the index $(a,b)$ rather than $(i,j)$ because this index is over the image itself and has a stride of one pixel. This whole operation amounts to a central difference scheme Wereley and Meinhart, 2000, and any disagreement between $I_{n+}$ and $I_{n-}$ represents residual error in $\underline{d}_n$ (plus presumably some small bias and noise).

At this point we have a displacement field estimate from the NVOFA which is robust but not sufficiently precise or accurate for PIV, and a set of images which have been each warped by half of this displacement field so that they should nearly match. From here, the processing is handled in much the same way as the standard PIV technique: Gaussian sub-pixel interpolation is performed on a 5-point correlation stencil about the shift which produces the maximum correlation between the two warped images (in this case the stencil's center is always chosen to be at zero-shift). That is, for correlation at a particular evaluation point $(i,j)$ with stride length $s$ between evaluation points,

$$c_{i,j}(k,l) \equiv \sum_{q=si}^{N-1} \sum_{r=sj}^{N-1} I_{n-}(q,r) I_{n+}(q+k, r+l), \quad (4.7)$$

where the correlation window has size $N \times N$, we can calculate the displacement correction, $\underline{\delta}_n(i,j)$, which maximizes the underlying continuous function $c_{i,j}$ by fitting a radially symmetric Gaussian to $c_{i,j}(k,l)$ centered about $k = l = 0$ using:

$$\underline{\delta}_n(i,j) = \begin{bmatrix} \frac{\ln[c_{i,j}(-1,0)] - \ln[c_{i,j}(1,0)]}{2\ln[c_{i,j}(-1,0)] - 4\ln[c_{i,j}(0,0)] + 2\ln[c_{i,j}(1,0)]} \\ \frac{\ln[c_{i,j}(0,-1)] - \ln[c_{i,j}(0,1)]}{2\ln[c_{i,j}(0,-1)] - 4\ln[c_{i,j}(0,0)] + 2\ln[c_{i,j}(0,1)]} \end{bmatrix}. \tag{4.8}$$

This yields a sub-pixel correction to $\underline{d}_n$ so long as $\underline{d}_n$ was itself within roughly 1 pixel of the true displacement: $\underline{v}'_n = \underline{d}_n + \underline{\delta}_n$. Since equation 4.8 only requires evaluation of $c_{i,j}(k,l)$ at 5 different points for each vector, we performed the cross correlation calculation at those points directly from definition 4.7. The data required for these calculations is almost identical, so we loaded the data necessary for the 5 cross correlations into GPU shared memory all-at-once.

Once $\underline{v}'_n$ is obtained, the universal outlier detector is optionally applied (Westerweel and Scarano, 2005), and outliers are replaced with the median of their non-outlier neighbors. This produces the penultimate intermediate displacement field, $\underline{\tilde{v}}_n$, from $\underline{v}'_n$. The last processing step is to perform exponential smoothing in time on $\underline{\tilde{v}}_n$. That is, $\underline{v}_{rapid_n} = \alpha_2 \underline{\tilde{v}}_n + (1 - \alpha_2)\underline{\tilde{v}}_{n-1}$. Finally, the displacement fields (and optionally the processed images) are optionally saved via transfer from GPU memory to non-volatile storage (such as hard-drive), and displayed when the screen refreshes.

## 4.3 Setup

In this section we describe the numerical and experimental setups used to compare our software to a baseline, and the parameters used by RapidPIV and the baseline.

RapidPIV was tested using two different physical experiments against one of the best commercially available multi-grid PIV software programs, PIVview from PIVTEC (we did not find the performance of free software sufficient to act as a baseline for these sequences). We chose a typical set of multi-grid parameters in PIVview as opposed to its most advanced features because the availability of advanced features varies between software programs. As a consequence, the results may not reflect the ultimate limit of what PIVview or its competitors are capable of, but we believe it indicates what a refined but standard multi-grid implementation is capable of. We therefore hereafter refer to the baseline as "multi-grid PIV."

### 4.3.1 Uniform Synthetic Flow

A uniform flow of synthetically generated particle images was used to precisely benchmark the performance of RapidPIV in an idealized and controlled setting,

and to determine how various PIV parameters affect this ideal performance. To do so, we generated $2048 \times 2048$ px 8-bit grey-scale images seeded with a total of $n_p$ additive radially symmetric Gaussian intensity distributions with equal intensity (such that the peak of the Gaussian is 127 counts) and diameter $d_p$ equal to 4 times the standard deviation of the Gaussian. These Gaussians represent the particles. The parameter $n_p$ furnishes the particle density $\rho_p = n_p/2048^2$. The position of each particle in the first image is a uniform random variable between 0 and 2047 in both components, $\underline{z}_p$, while the position in the second image is $\underline{z}_p + \begin{bmatrix} \Delta_p \\ 0 \end{bmatrix}$, so the size of the displacement is $\Delta_p$. Consistent with a top-hat laser sheet profile and uniform particle distribution, we assign a probability of $p$ that a particle which exists in the first frame disappears in the second, and with probability $p$ particles pop into existence in the second frame which were absent in the first. The images are corrupted by additive Gaussian white noise with standard deviation $\Lambda$ and mean $3\Lambda$. The noise can be normalized by the intensity of the particle images to obtain a measure of the noise-to-signal ratio: $\lambda = \Lambda/127$. A total of 5 experiments are performed in which a single parameter from the group $(\Delta_p, d_p, \rho_p, p, \lambda)$ is swept across a range of values with the other 4 parameters held fixed at their defaults. The defaults are $\Delta_p = 0.5$ px, $d_p = 3$ px, $\rho_p = 1/32$ ppp, $p = 0$, $\lambda = 4/127$. A single image is used per datapoint, and we ignore the 5 closest vectors to the edge of the velocity field so that the performance on the interior of the domain is being measured. This results in each data-point representing 13924 vectors, of which a quarter are statistically independent at the last stage of each algorithm (since in both cases the window size is $32 \times 32$ px with 50% overlap).

The images are processed by both RapidPIV and multi-grid PIV. The processing parameters used by both software are identical to the ones used in section 4.3.3 below, except for with RapidPIV we set $\alpha_1 = \alpha_2 = 1$.

### 4.3.2 Plate Accelerated From Rest

A sharp-edged solid plate of chord length $c = 61$ mm and span 315 mm held between acrylic end-plates was accelerated along its normal at a constant rate from rest to a velocity of $v_f = 0.075$ m/s through water ($Re \equiv \frac{cv_f}{\nu} = 4900$) in a distance of $c/4$. The plate held this constant velocity up to a formation time of $T \equiv \frac{1}{c}\int_0^t v(\tau)d\tau = 8.2$ before decelerating to rest. The flow was illuminated from behind at center-plane by a 450 nm 10 W laser expanded by a cylindrical lens. The plate is mostly transparent, except for the edges, so illumination is provided almost everywhere. The physical

setup is shown in figure 4.3. Seeding was provided by 13 $\mu$m nominal diameter Potters Industries silver coated glass micro-spheres. A total of 1000 images, each, were taken at 100 Hz via 3 synchronized IDT XSM-3520 High-Speed cameras aligned perpendicular to the laser plane. The images were unwarped and stitched together into a single $4999 \times 1391$ px image such that no occluded region exists throughout the plate's motion. Intensity matching across the stitched region was enforced by subtracting a Gaussian blurred version, with $\sigma = 10$ px, of the image from a slightly Gaussian blurred version of the original with $\sigma = 1$ px, and applying CLAHE with clipping limit of 5 bins. Since the stitching process already introduced the necessary pre-processing, and to ensure consistency between the quality of images presented to the software, further pre-processing was avoided by disabling this pipeline for both software. An example of what these stitched and processed images look like is shown in figure A.13 (the blue and yellow frames mark regions of interest used in figure 4.4.3). Particle concentration is roughly 0.01–0.02 ppp. An example raw image from the right camera is shown in figure A.12.

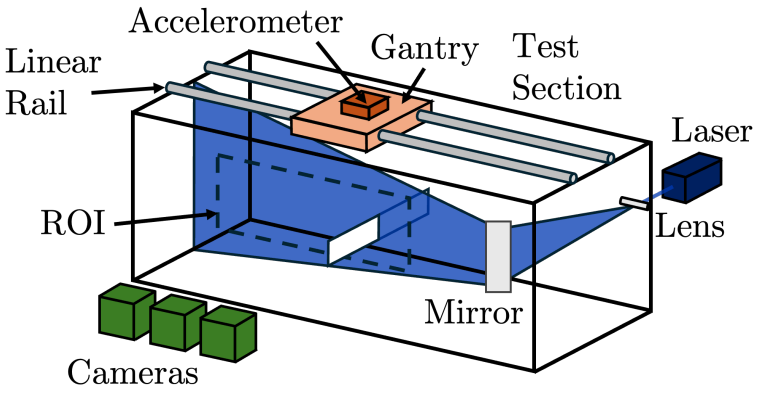


Figure 4.3: Solid plate experimental setup. Endplates connecting plate to gantry not shown.

The processing options used for the multi-grid PIV method were: 16 px final window size (uniformly weighted, $96 \times 96$ px initial size), and 50% overlap. Post-processing was provided by universal outlier detection (threshold 2, regularizer 0.1) and 25% correlation-coefficient thresholding. Outliers were replaced with secondary peaks or by interpolation, with 5 passes. All other settings were default. RapidPIV was evaluated using $16 \times 16$ px correlation windows and 50% overlap, with universal outlier detector threshold of 2, and universal outlier detector regularizer of 0.1 px. The exponential smoothing was $\alpha_1 = \alpha_2 = 0.5$.

This sequence was processed offline by RapidPIV. The flow contains a shear layer

emanating from the plate's edges whose initial thickness is below what the seeding density can resolve. It breaks up into small yet distinct and resolvable vortices via the Kelvin-Helmholtz instability. Most of the fluid is at rest, but the fluid in the Kaden spiral formed at startup is rotating and developing quickly early on, so velocity dynamic range and flow unsteadiness are high. Towards the end of the plate's motion the wake becomes turbulent and 3-dimensional. This makes loss of particle pairs substantial.

### 4.3.3 Cylinder Wake

A cylinder of diameter $d = 5.9\,\text{mm}$ and span $w = 130\,\text{mm}$ was held between two end plates. The flow was illuminated at center-plane from below by the same laser. A flow of water ($Re \equiv \frac{u_\infty d}{\nu} = 1300$) with velocity $u_\infty = 0.020\,\text{m/s}$ was passed over it, as shown in figure 4.4 (for more information on the water channel used for this experiment, see Devey and Gharib, 2025). An Allied Vision Alvium U-158m camera was focused on the wake of the cylinder and took samples at 250 fps. A total of 2500 image samples were acquired, with each image being $1456 \times 544\,\text{px}$. The particle concentration was roughly 0.002 ppp, and the concentration of bright particles was substantially lower (figure A.14 shows an example raw image).

Because of the low particle concentration, high displacement, and large gradients, resolving this flow is a challenge. Again, to ensure consistency of pre-processing, both image sequences were pre-processed in the same way before being sent to the software by subtracting a Gaussian blurred version, with $\sigma = 10\,\text{px}$, of the image from the original (high-pass filter), and enhanced using CLAHE with clipping limit of 5 bins (the processed example image is shown in figure A.15). Further pre-processing was avoided by disabling this pipeline for both software. Reference velocity fields were again provided by multi-grid PIV with the same options as in section 4.3.2 except the final window size was $32 \times 32\,\text{px}$ and the initial window size was $160 \times 160\,\text{px}$. RapidPIV was evaluated at $32 \times 32\,\text{px}$ resolution with 50% overlap, with universal outlier detector threshold of 2, and universal outlier detector regularizer of 0.1 px. The exponential smoothing was $\alpha_1 = \alpha_2 = 0.5$.

## 4.4 Results & Discussion

### 4.4.1 Compute Performance

The compute performance of RapidPIV was tested on image sequences of various sizes up to the maximum it can handle due to the hardware limit set by the NVOFA: 8192×4096 px, or 33.6 megapixels (MP). A laptop with Nvidia RTX 3050 Mobile Ti

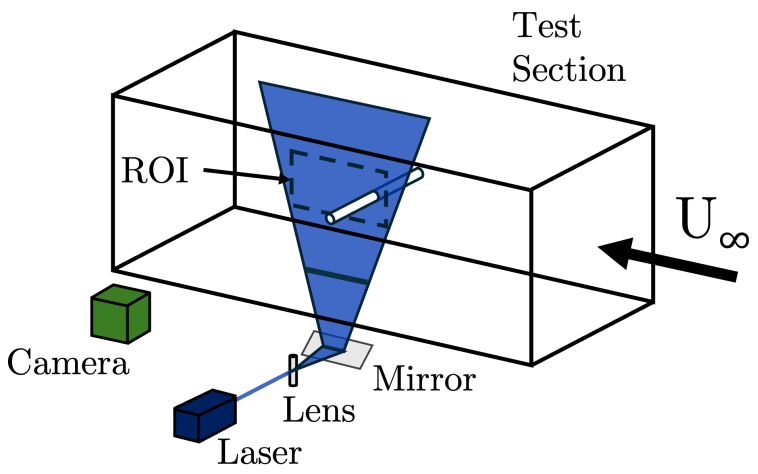


Figure 4.4: Cylinder experimental setup. Endplates not shown.

GPU (Ampere generation), and a desktop with Nvidia RTX 4080 (Ada generation) were used for the tests. The results, showing the number of full-density velocity field calculated per second at 50% overlap and $16 \times 16$ px window size, are shown in figure 4.5. The frame rate shown is the number of frame pairs processed per second, so if frame-straddling is used the camera's video capture rate would have to be twice the stated number to keep up with the computer. A few key trends emerge. First, both hardware options tested exceed maximum frame rate capabilities demonstrated in the literature: on small but non-trivial image sizes >2000FPS is seen with RapidPIV whereas the highest values previously reported for dense velocity fields was roughly 1000FPS by Pimienta and Aider, 2024. Second, for sufficiently small images the number of frames calculated per second (FPS) is independent of image size on the RTX 4080. We believe this is because for sufficiently small images the program becomes overhead limited. For larger images, roughly 1MP or more, the FPS becomes throughput limited; the frame-rate is inversely proportional to the size of the image. Indeed, for larger images RapidPIV's calculation speed can be calculated from the image size as,

$$\text{FPS} = (\text{MP}) \cdot (\text{Throughput}), \tag{4.9}$$

where Throughput = 1150MP/s for the RTX 4080, and Throughput = 460MP/s on the RTX 3050 Mobile Ti ($\approx$ 18 million vecs/sec and 7 million vecs/sec, respectively). In Pimienta and Aider, 2024 throughput-limited performance on the RTX 3090 desktop GPU was almost identical to what we saw on our RTX 3050 Mobile Ti laptop GPU, at 480MP/s, using the FOLKI PIV algorithm Champagnat, Plyer, Le Besnerais, Leclaire, Davoust, and Le Sant, 2011. The highest throughput-limited

performance demonstrated in the literature previously was 816MP/s on a Titan RTX by Wernet, 2019 for the purpose of background oriented schlieren (the algorithm they used is structurally the same as single-pass cross-correlation PIV), which is still somewhat below what we measured on the RTX 4080. Since they used an older GPU and a much larger $64 \times 64$ px window size it is not clear if their software would be faster or slower than that if run on an RTX 4080 with $16 \times 16$ px windows, as we do. They showed their software's performance is roughly proportional to the GPU's memory bandwidth, of which the RTX 4080 has 7% more, but the smaller window size increases the number of vectors which need to be computed by a factor of 16, which may slow it down. It is also worth pointing out that both of these algorithms were optimized purely for speed (ie single-pass in the case of Wernet, 2019). The software which is commonly used by practitioners outside the context of RTPIV is more robust but slower. For example, PIVview has a throughput limited performance of 1.73MP/s on the same desktop we used to run RapidPIV, 665 times slower on the same desktop (the multi-grid PIV software uses the desktop's Intel i5-13600KF rather than the RTX 4080).

The NVOFA is a separate resource, within the GPU, from the Streaming-Multiprocessors (SM's) which perform pre and post-processing. Figure 4.6 shows, to scale, what GPU resources are used at each stage of the RapidPIV pipeline for 3 successive frames. Latency of each frame is slightly more than what this figure would indicate, as this timing configuration is only possible if the camera and RapidPIV throughputs match exactly. As can be seen, the NVOFA is the limiting factor for the throughput. While the NVOFA calculates flow vectors, RapidPIV executes pre- and post-processing on the streaming multiprocessors (SMs), an area of the GPU independent from the NVOFA. Because the pre and post-processing together take less time than the NVOFA takes to run, the pre and post-processing time do not affect throughput, only latency. This also means that the SM's are idle most of the time. There is therefore an intriguing possibility of using RapidPIV concurrently on the same GPU with other intensive processes, such as machine learning, or running more advanced pre or post-processing algorithms without performance tradeoffs.

On mobile devices, such as laptops or Nvidia Jetson modules, GPU power constraints can cause throttling and reduce throughput. For example, on a Dell XPS 9510 laptop, the RTX 3050 Ti Mobile GPU is limited to 40W, which throttles RapidPIV to 81% NVOFA utilization when running the full processing pipeline. On the desktop machine with the RTX 4080, power consumption is 90W, far below its

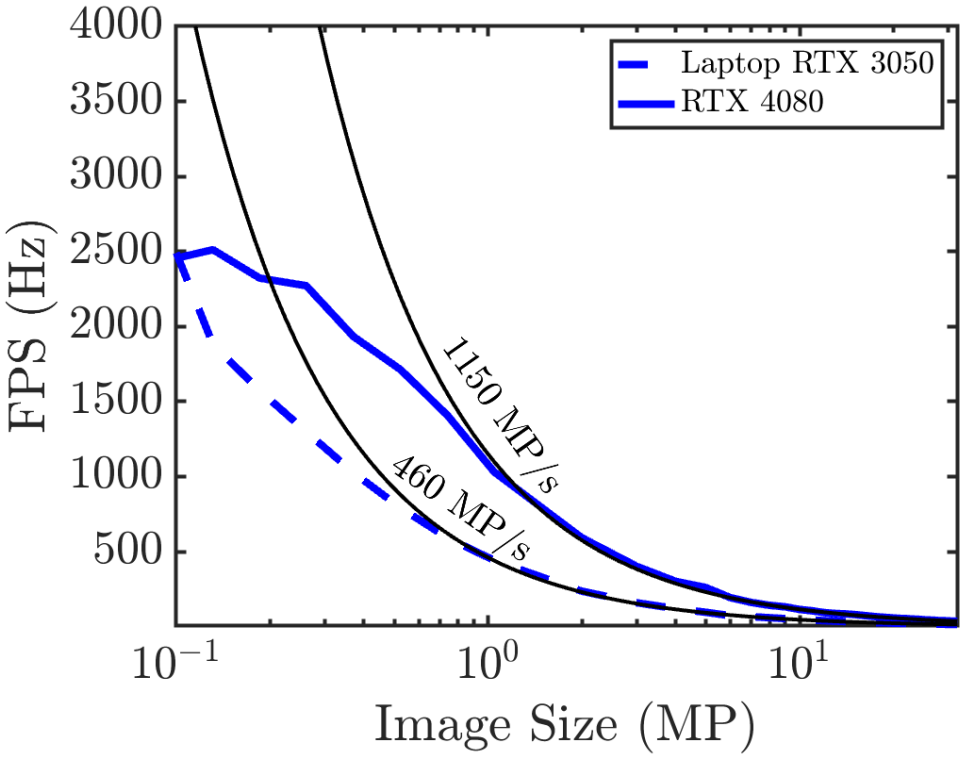


Figure 4.5: Log-linear plot of RapidPIV processing speed with Ada-generation (RTX 4080; solid blue line) and Ampere-generation (RTX 3050 Mobile Ti; dashed blue line) GPUs. Performance is measured in frames per second against the size of the image (in megapixels).

throttling limit of 320W. As a result, RapidPIV achieves 97% NVOFA utilization. These differences contribute in part to the substantial performance difference seen in figure 4.5 between these two pieces of hardware.

There are also possibilities associated with RapidPIV's compression capabilities, because when its source is a camera's video stream it becomes optional to save the raw image files. Processed velocity field `.mat` files of 1 MP images take 17.7 kB storage on average, at 50% overlap and 16 px window size, a 57× compression ratio (compared to the 1MB raw image files). In cases where it is acceptable to not store the raw images, this compression ratio enables continuous high-speed streaming of vector fields to hard-drive, side-stepping the RAM limitation on experiment run time of standard high-speed cameras (up to the throughput limit).

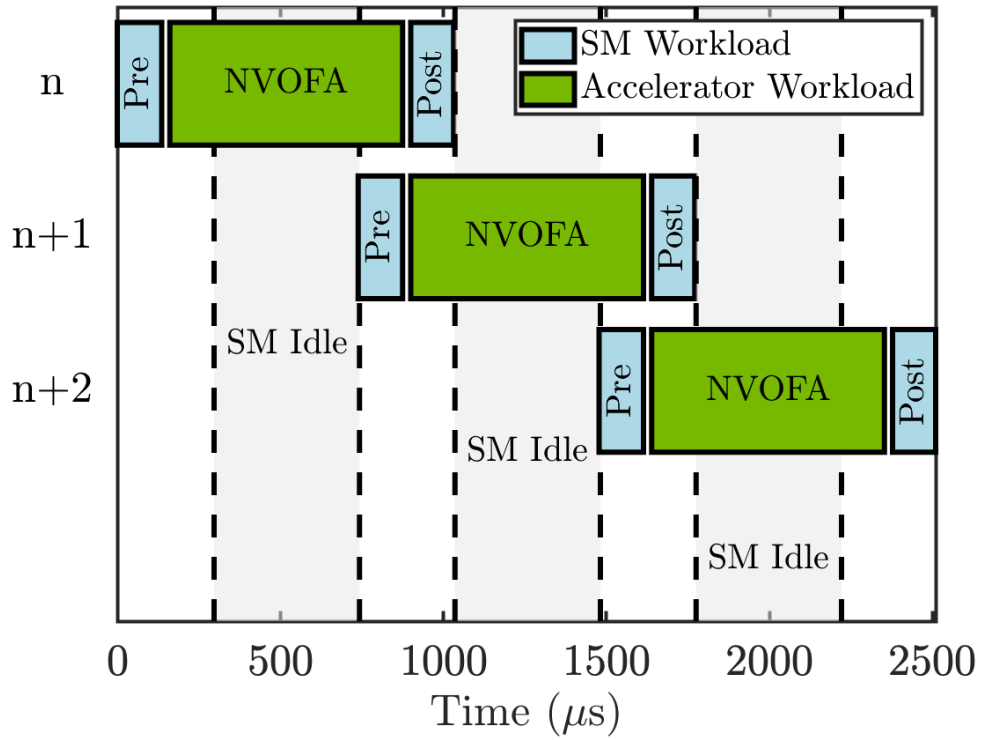


Figure 4.6: GPU Processing time breakdown for 1 MP images on RTX 4080. Figure is to scale. The pre and post-processing SM workloads have been each lumped to enhance readability.

### 4.4.2 Uniform Synthetic Flow Results

In all of the synthetic flow tests we track two quantities: the mean absolute error and standard deviation of the x-component of the velocity field. These quantities track the accuracy and precision of this velocity component, respectively. The mean absolute error for the RapidPIV velocity field's x-component, $u_{Rapid} = \underline{v}_{Rapid} \cdot \hat{\underline{e}}_1$, is calculated using,

$$\mu_{Rapid} \equiv \frac{\sum_{\forall(i,j)} u_{Rapid}(i,j)}{\sum_{\forall(i,j)} 1},$$
$$E_{Rapid} = \left|\mu_{Rapid} - \Delta_p\right|, \tag{4.10}$$

while the standard deviation is calculated using,

$$\sigma_{Rapid} = \sqrt{\frac{\sum_{\forall(i,j)} \left(u_{Rapid}(i,j) - \mu_{Rapid}\right)^2}{\left(\sum_{\forall(i,j)} 1\right) - 1}}. \tag{4.11}$$

The value $E_{multi}$ and $\sigma_{multi}$ for multi-pass PIV are calculated in the same manner.

In the first synthetic test we vary $\Delta_p$ from 0 to 4. As can be seen in figure 4.7, multi-grid is more precise and has least bias, with RapidPIV roughly a factor of 3 worse on both counts. Some degree of peak-locking is apparent in both software as their error is lowest for integer values of $\Delta_p$.

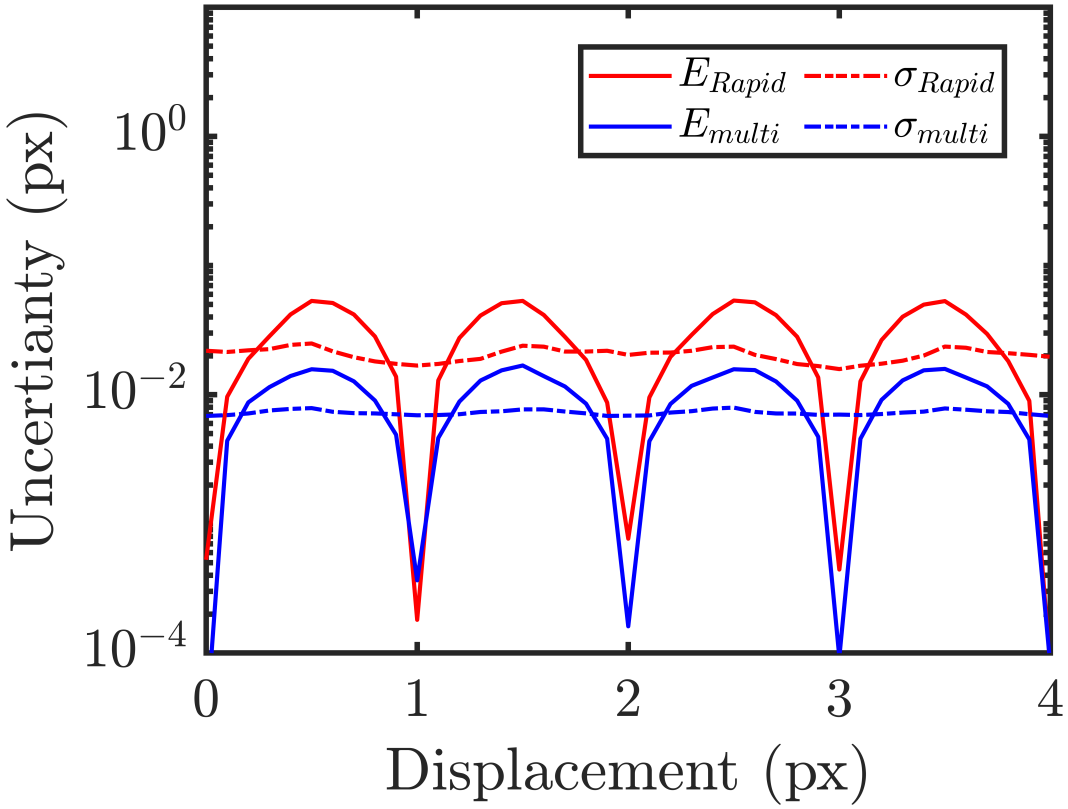


Figure 4.7: Effect of displacement magnitude, in a uniform synthetic flow field, on error magnitude (−) and standard deviation (−·).

Next, we vary the particle diameter from 0.5 to 8 px, shown in figure 4.8. For $d_p < 2$ px both error and standard deviation are high for both software, which is a known fundamental limitation of digital PIV that in real situations is related to the camera's pixel geometry Westerweel, 2000a. Multi-grid PIV shows minima in both the error magnitude and standard deviation at 3–4 px particle diameter whereas RapidPIV's standard deviation is invariant to particle diameter and its error magnitude improves with increasing particle diameter (although presumably the precision would worsen if diameter became even larger due to fundamental statistical limits Westerweel, 2000b). The multi-grid method uses bicubic interpolation for its image warping, whereas RapidPIV uses bilinear interpolation. The known improvement in bias for small particle diameters associated with bicubic interpolation over bilinear interpolation could explain the observed differences in error magnitude between

these two software, both in the particle diameter and particle displacement tests, at least when using synthetic data (see Astarita and Cardone, 2004).

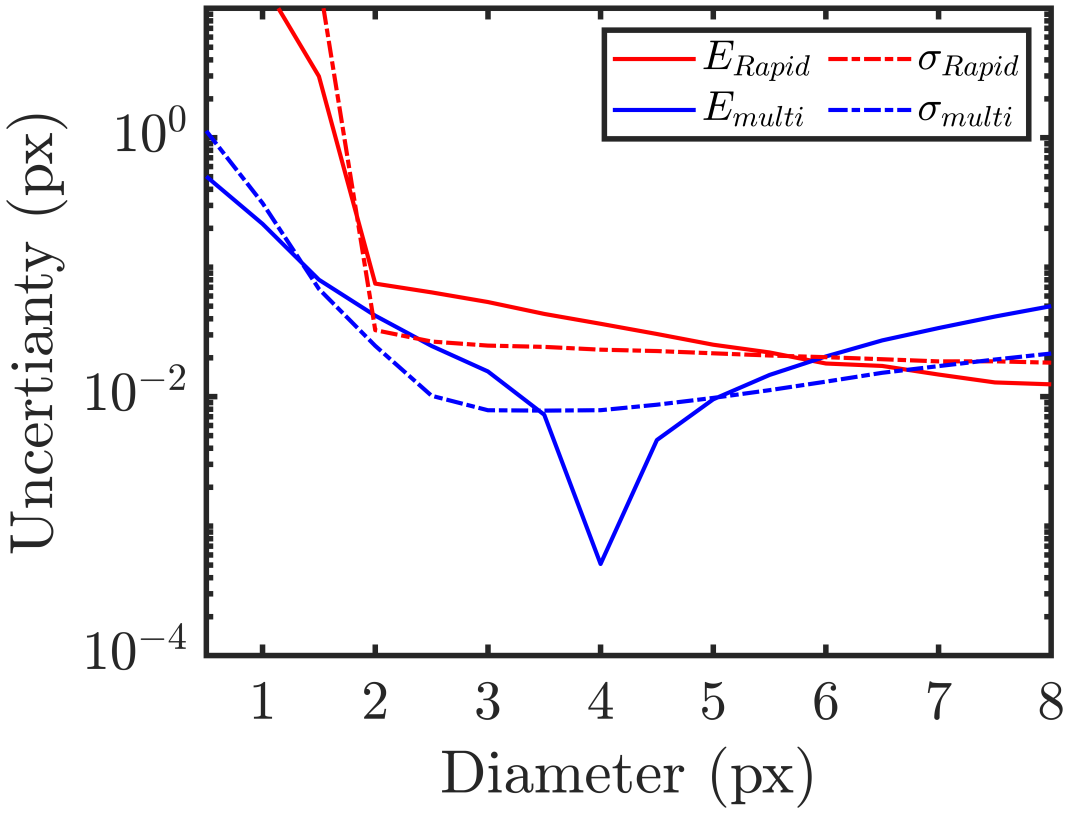


Figure 4.8: Effect of particle diameter, in a uniform synthetic flow field, on error magnitude (−) and standard deviation (−·).

We also varied the particle seeding density from $3.2 \cdot 10^{-3}$ ppp to 0.2 ppp, shown in figure 4.9. Over this range of densities, large-velocity outliers (ie $>$ 4 px displacement from the mean) were very rare and thus excluded. Both RapidPIV and multi-grid PIV have invariant error magnitude for roughly $\rho_p > 3 \cdot 10^{-3}$ ppp, but below this level there are too few particles in a window to get reliable velocity fields. Increasing seeding density improves precision in both software, with the gap in performance between the two shrinking as seeding density increases.

We varied $p$, the probability of a particle being lost or gained out-of-plane between the images (so $1 - p$ is pair correspondence fraction) from 0 to 0.5, shown in figure 4.10. For $p \leq 0.1$ both methods behave similarly, with invariant error and increasing standard deviation. However, beyond this point the performance of RapidPIV rapidly degrades whereas multi-grid PIV loses precision more gracefully. For $p > 0.15$ RapidPIV produces too many outliers, so the precision and bias are

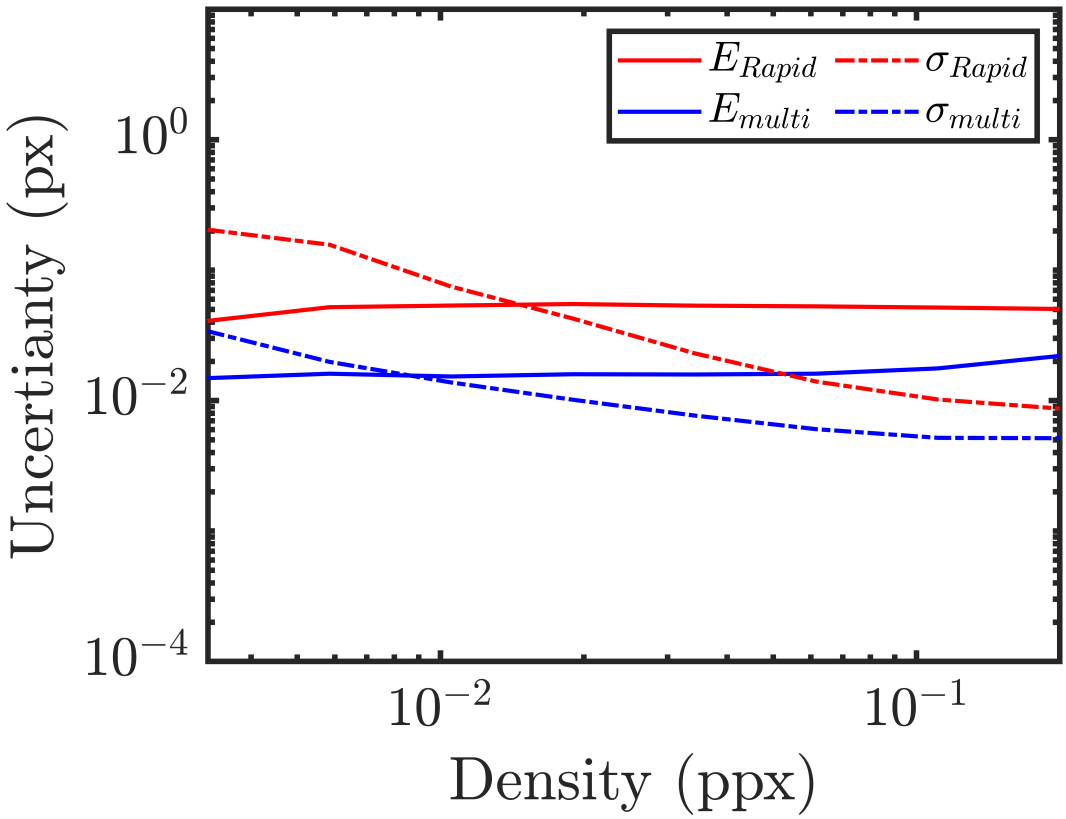


Figure 4.9: Effect of particle seeding density, in a uniform synthetic flow field, on error magnitude (−) and standard deviation (−·).

not shown. Correlation-based PIV is known to produce few outliers up to roughly $p = 0.6$ in these sorts of situations Nobach and Bodenschatz, 2009, so if the primary flow component is out-of-plane this could be a limitation of RapidPIV in comparison to multi-grid PIV. It should be pointed out, however, that this is a worst-case scenario for RapidPIV because both the NVOFA and our processing pipeline use information about the previous vector field as well as neighboring regions to avoid producing outliers. In these tests we only processed a single image pair and the out-of-plane motion was uniformly high, making these techniques ineffective. We do not observe substantial outliers due to out-of-plane effects in either of our physical wake experiments (see below), although this would likely become an issue for stereo PIV setups if the primary flow component were not in-plane.

Finally, we varied the amount of white noise in the images from $\lambda = 0$ to $\lambda = 40/127$, the results of which are in figure 4.11. Until the noise exceeds 0.15, the error magnitude is unaffected, and the precision of RapidPIV becomes closer to that of multi-grid PIV as their precisions both worsen steadily. Beyond this point multi-grid PIV continues the same trend but RapidPIV's accuracy and precision quickly

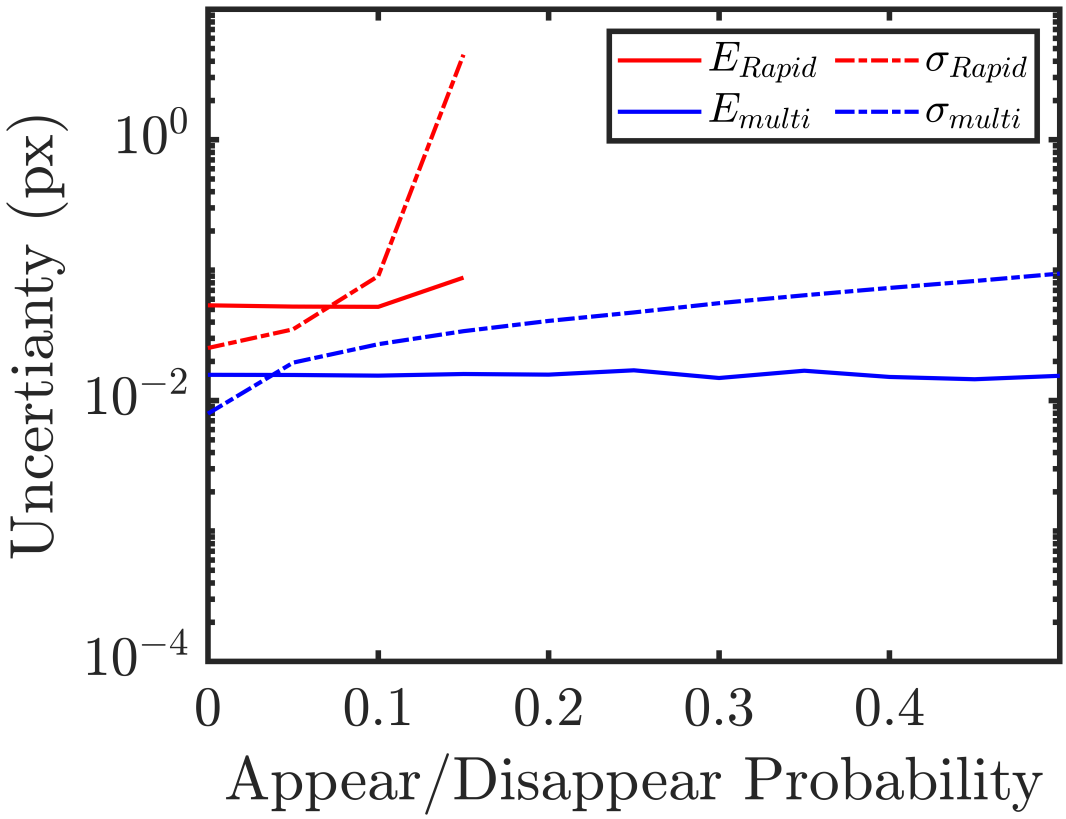


Figure 4.10: Effect of particle appearance/disappearance probability (or out-of-plane motion for top-hat laser sheet), in a uniform synthetic flow field, on error magnitude (−) and standard deviation (−·).

degrade. Beyond 0.2 there are too many outliers, so we do not show the precision or bias. On modern CMOS cameras this is unlikely to be an issue unless light is severely limited.

### 4.4.3 Accelerated Plate Results

The plate impulsive start experimental data showed a generally excellent agreement between RapidPIV and multi-grid PIV. Consider first the probability density functions (pdf's) of the plate-normal fluid displacement between frames as measured by the raw NVOFA, RapidPIV, and multi-grid PIV: $f(\underline{v}_{NVOFA} \cdot \hat{e_1})$, $f(\underline{v}_{Rapid} \cdot \hat{e_1})$, $f(\underline{v}_{multi} \cdot \hat{e_1})$. These probability densities are shown together in figure 4.12. It is clear from $f(\underline{v}_{NVOFA} \cdot \hat{e_1})$ that the raw NVOFA is insufficient for most PIV applications: peak locking and discretization error are present. Meanwhile, $f(\underline{v}_{Rapid} \cdot \hat{e_1})$ is the same as $f(\underline{v}_{multi} \cdot \hat{e_1})$ aside from slightly lower noise (narrower pdf around 0 matching the largely quiescent nature of the flow field) at the cost of slight under-prediction of the velocity tails in log-scale. Figure 4.14 depicts the noise in more

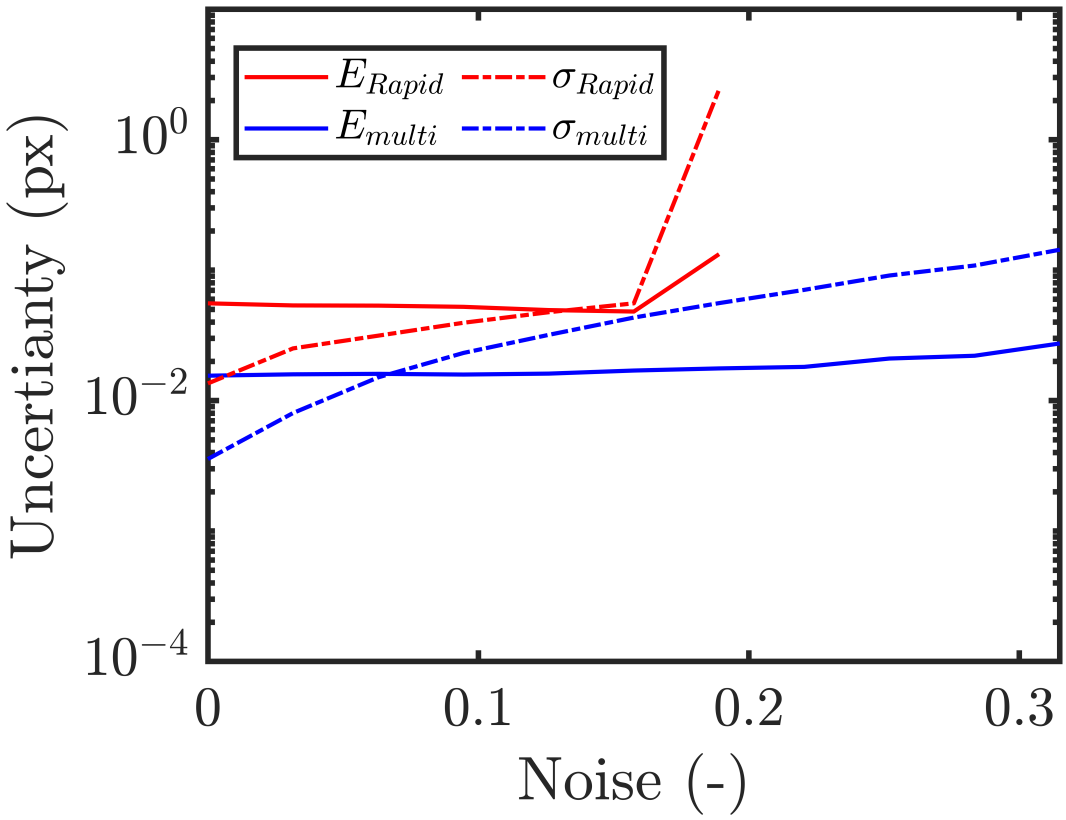


Figure 4.11: Effect of Gaussian white noise, in a uniform synthetic flow field, on error magnitude (−) and standard deviation (−·).

detail by showing the probability density of the plate-normal flow velocity throughout the fluid in the region highlighted by the blue frame in figure A.13. It shows that the background turbulence intensity in the nominally "quiescent" fluid results in a roughly 0.1 px-per-frame standard deviation in particle displacement (black curve). The distribution of the displacement rate as measured by the raw NVOFA (green curve), RapidPIV (red curve), and multi-grid PIV (blue curve) all mostly capture this background turbulence intensity level. However, RapidPIV comes the closest followed by multi-grid and finally the raw NVOFA (the NVOFA's wider bins reflect its $1/32$ px displacement resolution). Now consider the discrepancy field between RapidPIV and multi-grid PIV,

$$D \equiv \underline{v}_{Rapid} - \underline{v}_{multi}. \tag{4.12}$$

Figure 4.13 shows that $D < 0.2$ px most places in the flow, which according to figure 4.14 is consistent with the combined random errors of multi-grid PIV and RapidPIV. However, RapidPIV appears to apply slightly more spatial smoothing to the vorticity field than multi-grid PIV does. Partially for this reason, $D$ is higher in

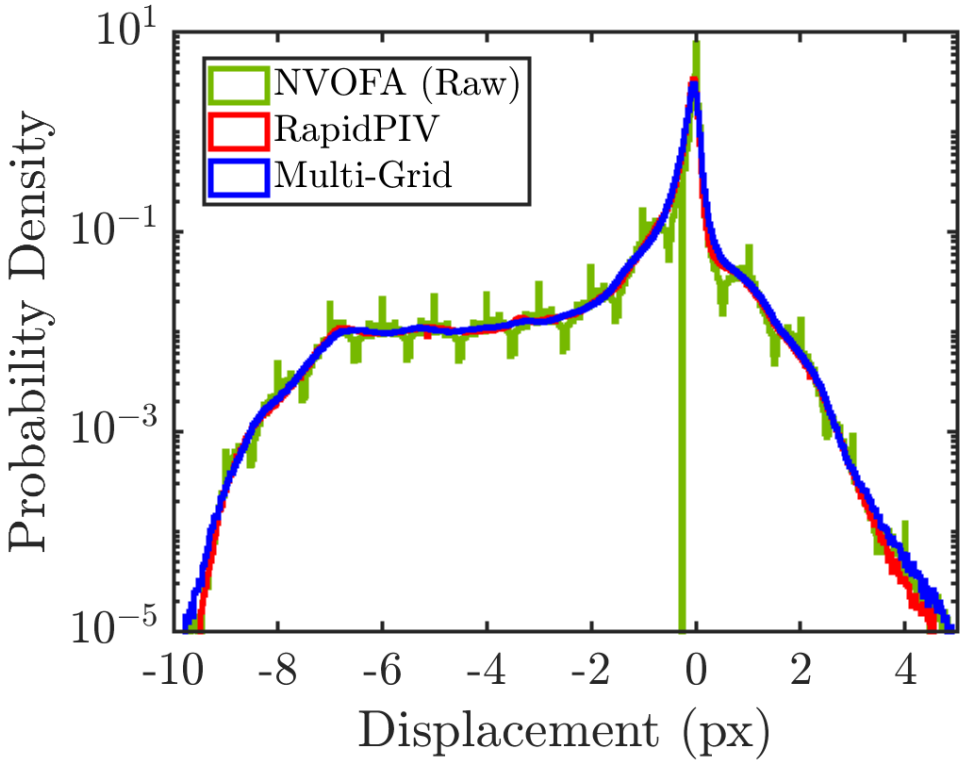


Figure 4.12: Probability density of the plate-normal flow velocity throughout the flow and throughout the experiment.

highly vortical regions. However, this is not an indication per-se that RapidPIV is under-performing. Indeed, close inspection of the vorticity field at $T = 4$ indicates what appears to be more artifacts in the multi-grid PIV field than the RapidPIV field. In particular, the upper starting vortex as measured via multi-grid PIV has a region of near-zero vorticity just off-center, while the lower one has an unusually sharp vorticity profile at the center. These artifacts are fairly confined, but they do not appear in RapidPIV, nor do they appear when multi-grid PIV is used with the advanced feature, correlation-plane summing (not shown). The relationship between $D$ and the magnitude of vorticity, $|\omega|$, can be represented by the conditional probability of $D$ given $|\omega|$: $f\left(D \middle| |\omega|\right)$. This conditional as a function of $|\omega|$, along with the expectation value of $D$ given $|\omega|$,

$$E\left[D \middle| |\omega|\right] \equiv \int_0^\infty D f\left(D \middle| |\omega|\right) dD, \tag{4.13}$$

is shown in figure 4.15. For small $|\omega|$ discrepancy is small and independent of $|\omega|$, indicating the discrepancy here is explained by their combined noise-floors. However, for $|\omega| > 10^{-2}$ stronger vorticity tends to correlate with larger discrepancies,

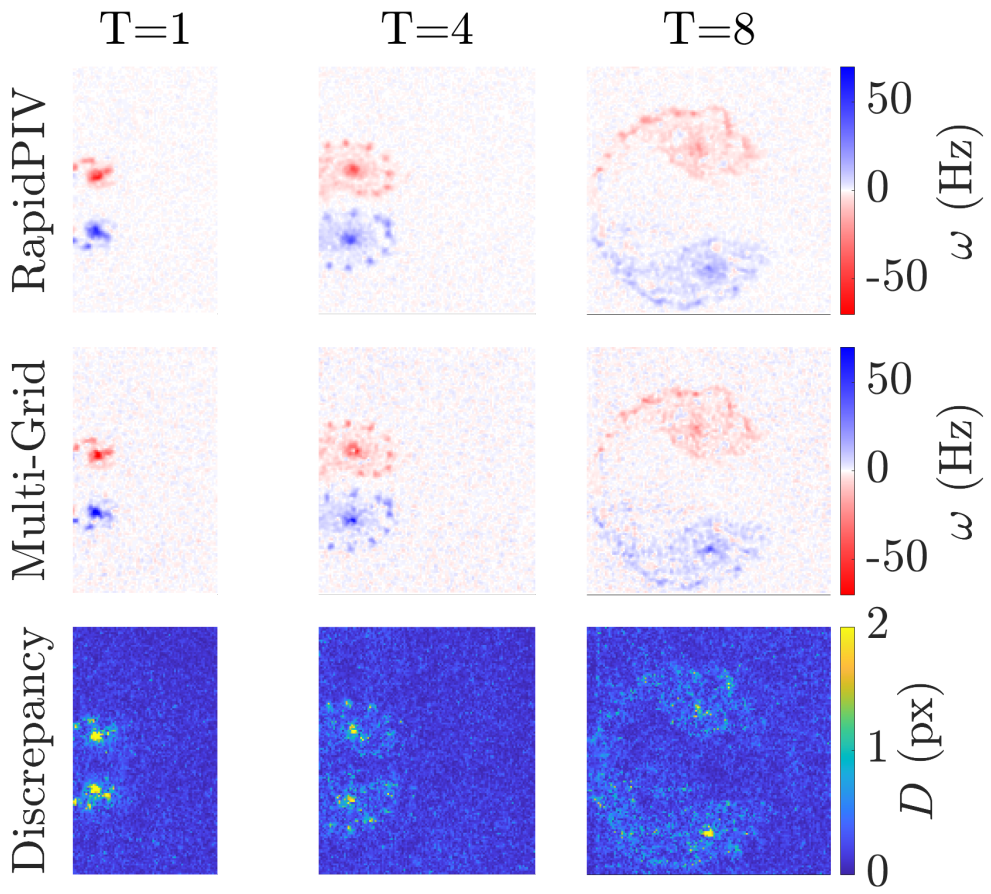


Figure 4.13: *Upper and middle sub-plots:* Comparison of the vorticity field computed by RapidPIV (upper sub-plots) and multi-grid PIV (middle sub-plots) in the wake of a flat plate at various non-dimensional formation times, $T = \{1, 4, 8\}$. *Bottom sub-plots:* Velocity discrepancy between multi-grid PIV and RapidPIV at the same set of formation times as the sub-plots above.

indicating one or both of the software develop bias in highly vortical regions, as expected. For the largest vorticities (which are rare, hence the higher noise in this region of the PDF) discrepancies can exceed 2 pixels (the max discrepancy across the entire field and the entire sequence is 5 px). The vorticity-invariant low vorticity magnitude behavior of $f\left(D \middle| |\omega|\right)$ suggests a noise floor. To compare the random error of the raw NVOFA, multi-grid PIV, and RapidPIV, the pdf of displacement components was calculated for each of these methods in a well illuminated region of the fluid behind the plate, for the first 100 frames (before the plate was set in motion). The result of this calculation is displayed in figure 4.14. The background flow was determined using the pdf of the temporal mean of the multi-grid PIV velocity field

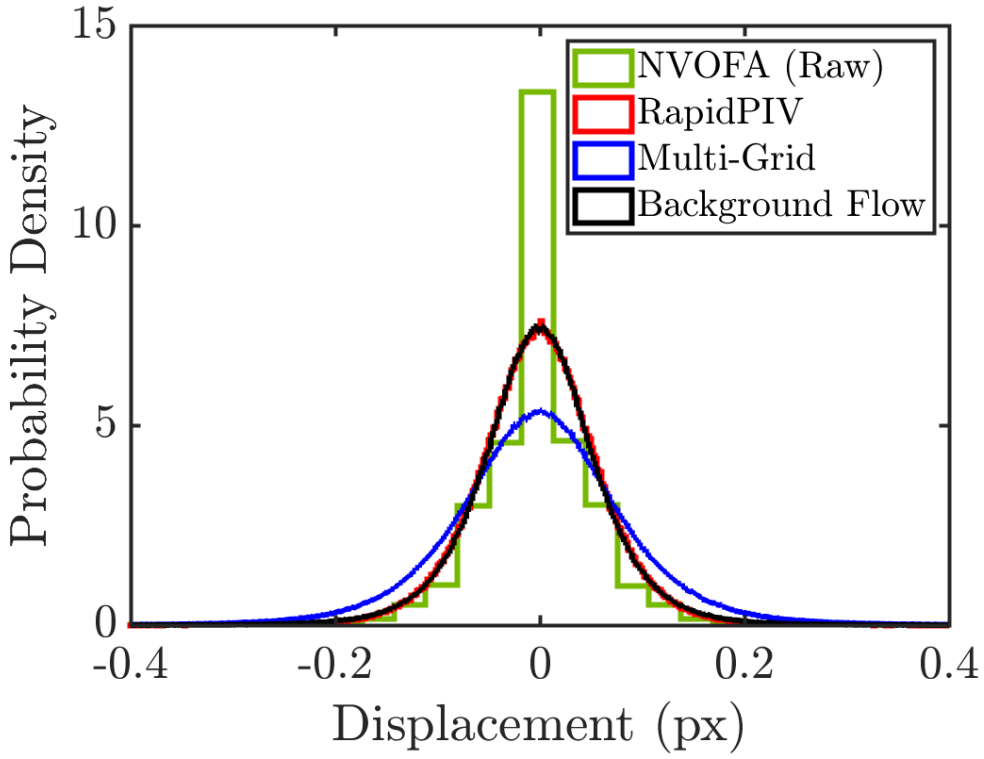


Figure 4.14: Probability density function of the image displacement between frames, before the plate begins motion.

[2]. The fluid itself was not perfectly quiescent, so an ideal histogram should match the one of the background flow, not be a delta function. The NVOFA histogram peak-locked onto 0 so it under-predicted the turbulence intensity. RapidPIV slightly over-predicted turbulence intensity due to random error, while the larger random error of multi-grid PIV resulted in a larger over-prediction. This supports our previous observation that RapidPIV produces lower noise because it slightly spatially smooths the velocity fields (which is undesirable), and we are doing a small amount of temporal filtering via the exponential average.

### 4.4.4 Cylinder Wake Results

The cylinder wake experiment presented a challenge to multi-grid PIV whereas RapidPIV could reliably obtain velocity fields at a window size of $32 \times 32$ px (except at the edges of the flow where neither method performed because of out-of-frame

[2]The flow varied little during those 100 frames because the flow velocity was order 1–2 mm/s while the energetic structures in the flow were on the order of multiple cm. It was verified that this meant averaging to reduce PIV random noise would not under-represent background flow intensity by using different averaging intervals. The distribution appeared to asymptote as the interval was increased.

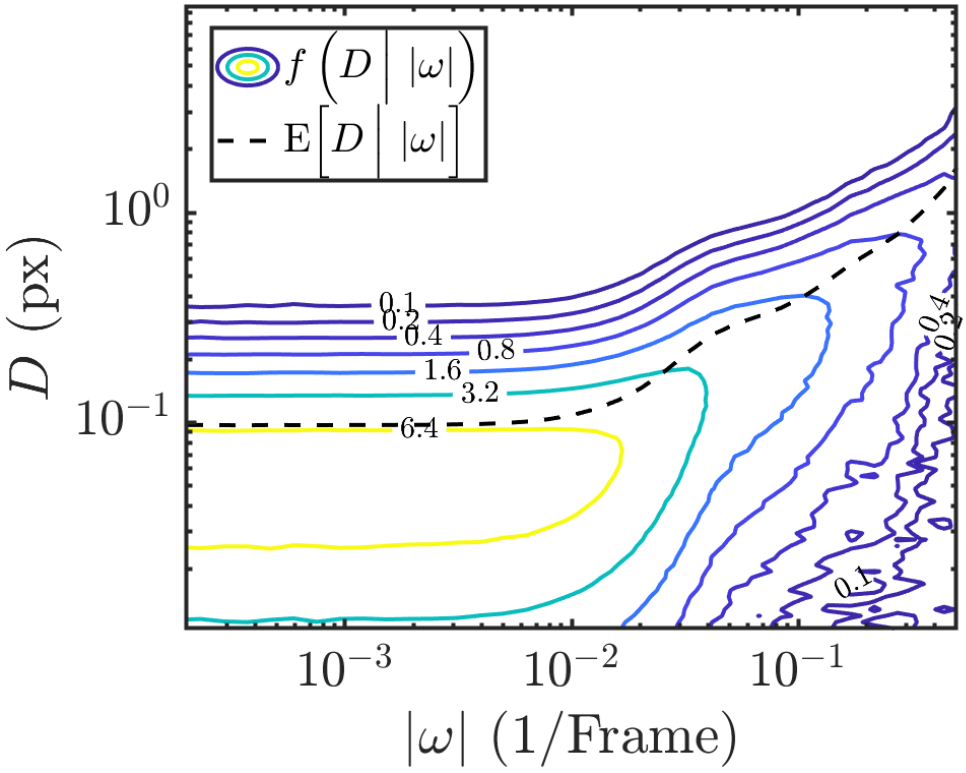


Figure 4.15: Log-log contour plot, with logarithmically spaced level sets, of the conditional probability distribution of the discrepancy between RapidPIV and multi-grid PIV conditioned on the magnitude of vorticity, as a function of the magnitude of vorticity. The expectation value of $D$ given $|\omega|$ is also shown (black $--$).

motion, not shown). The sequence is really a better candidate for PTV than PIV since the seeding density is so low compared to the size of the velocity gradients, however it provides an interesting point of comparison for how the two methods tackle the problem of velocity estimation. Figure 4.16 shows the vertical velocity in the wake of the cylinder, as measured by RapidPIV and multi-grid PIV, at two frames tightly spaced in time. The frames from RapidPIV are largely the same, but artifacts are present throughout the domain in the multi-grid PIV velocity field in both frames. To avoid these artifacts in multi-grid PIV a lower final resolution would be required. The place where RapidPIV performs poorly is the left edge of the velocity field, which is visible in frame 3. Since the SGM method builds a loss function inward from the edges of an image (see section 4.2.1 for more details), when there is low seeding density artifacts can appear in a thin region at the edge of images (2-3 grid units in this case), especially when the flow carries particles across the edge in question.

The flow between the time intervals in figure 4.16 should be nearly the same because the time intervals are spaced far enough that the temporal filtering in RapidPIV cannot explain the higher level of consistency in the RapidPIV frames compared to the multi-grid PIV frames. This is because the exponential smoothing performed in RapidPIV with our chosen $\alpha$ means data in frame 3 constitutes just 1/16th contribution to the data in frame 6. On the other hand, the frames are spaced close enough that the flow has not evolved appreciably since 3 frames represents 1/120th of a shedding cycle.

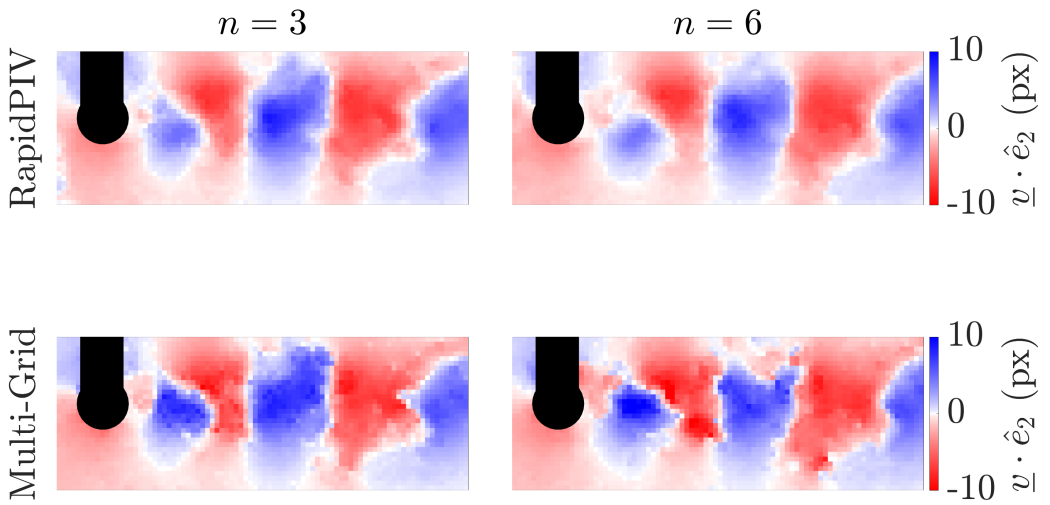


Figure 4.16: Comparison of the vertical velocity in the Von Kármán shedding wake of a circular cylinder at two near-by times (frame 3, and frame 6) as calculated using RapidPIV and multi-grid PIV. Occluded region is masked black.

To understand why RapidPIV outperforms multi-grid PIV in this scenario, we must look to the correlation plane. Figure 4.17 shows the correlation plane extracted from multi-grid PIV at a point where the velocity vector is an outlier. There are many equally good candidates for the correct flow displacement. If by chance the correct peak is not the highest one, as is the case here, an outlier results from exhaustive search of the correlation plane. Even though RapidPIV also relies on correlation, the peak which gets sub-pixel approximated is determined apriori by the NVOFA, not by exhaustive search. If the NVOFA provides an approximation which is close enough, the 5-point correlation stencil will refine the true peak rather than a spurious one. In that case, the fact that there are other equally high correlation peaks elsewhere in the correlation plane is of no consequence. There are so few particles per flow structure that the larger scale velocity fields used to refine the search at smaller scales, in the

multi-grid method, are not useful at the smaller scales. This dataset is therefore likely a better candidate for particle tracking, but the fact that the NVOFA can use neighboring regions to interpolate gracefully where there is missing data helps it perform outside the realm of where PIV can usually be applied.

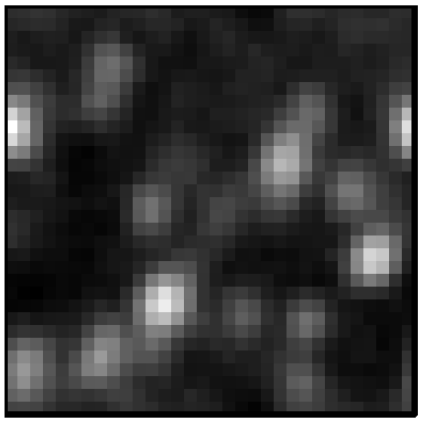

Figure 4.17: Correlation plane from multi-grid PIV at a point where the vector is an outlier. Note the multiple peaks.

## 4.5 Conclusions and Future Work

We developed an architecture which enables high-speed PIV processing on GPUs. This architecture is packaged into a flexible, PIV processing software with real-time velocity display capability. It is capable of either processing pre-existing image sequences, or continuously streaming from GenICam cameras, into non-volatile memory (i.e., the hard-drive). This was achieved by utilizing Nvidia's optical flow hardware accelerator to quickly produce a velocity field estimate. This estimate, while somewhat crude, is sufficiently close to the true velocity field that we were able to use classical PIV techniques to refine it. The precision, accuracy, and robustness of the system were tested on synthetic and experimental data. Precision and accuracy were found to often be comparable to a standard multi-grid PIV technique as provided by commercially available software. In our physical experiments we found that in highly vortical regions both software begin to fail in different ways. Robustness was often similar, but the cylinder experiment suggests that RapidPIV can perform better than multi-grid PIV in cases where particle tracking is better suited than both. On the other hand, our synthetic flow experiments showed that RapidPIV is more sensitive to high out-of-plane motion and to noise. This could present challenges to stereo applications or very light-limited experiments, respectively.

The processing speed of RapidPIV is 665 times faster than PIVview (on the same

desktop, although the multi-grid code does not take advantage of the GPU). When only the velocity fields are saved, just 1/57th of the storage space is required compared to saving the raw images. Since RapidPIV is compatible with streaming cameras this invites the possibility of continuous experiment recording. These properties could allow RapidPIV to expand the uses of PIV into areas where it was previously too slow, expensive, or impractical. For instance: biological or turbulence experiments which require exceedingly long run times at high frame rates, exploratory PIV-based flow visualization, or PIV for freely configurable scalar sensing. In the avian experiment mentioned in the introduction (see Hao, 2025), RapidPIV would allow each dataset to be processed in just 40 seconds rather than 7 hours, allowing data processing to keep up with the experiments so that bird handling rather than PIV processing is the most time consuming step. This would allow rapid experimental iterations which are currently impossible. The technique of combining optical flow hardware-accelerators with classical PIV algorithms may also yield advancements in RTPIV for control applications, as it both greatly improves throughput, and leaves plenty of GPU resources free for other uses. These resources could be utilized to simultaneously execute computationally intensive control schemes, such as those based on machine learning. Low-power optical flow accelerators exist, which could enable onboard PIV applications.