Proceedings of The Canadian Society for Mechanical Engineering International Congress 2018
CSME International Congress 2018
May 27-30, 2018, Toronto, ON, Canada

PARTICLE IMAGE VELOCIMETRY DATA PROCESSING ON A GPU CLUSTER

C. Dallas
Department of Mechanical and Industrial Engineering
University of Toronto
Toronto, Canada

P.E. Sullivan
Department of Mechanical and Industrial Engineering
University of Toronto
Toronto, Canada

***Abstract*—** Particle image velocimetry (PIV) data processing is a computationally expensive process. The immense time taken to analyze data can limit the maximum dataset size. Using graphics processing units (GPUs) has been shown to drastically decrease the processing time for PIV image pairs. The open-source PIV data processing software OpenPIV has been ported to run on a GPU to boost speed and efficiency and has outperformed the CPU version of the software. A multi-pass method is being implemented in OpenPIV to improve both speed and accuracy. The completed algorithm will be tested on an embedder CPU-GPU device, a desktop computer, and the SOSCIP GPU-accelerated supercomputing cluster. Ultimately, OpenPIV will run on a wide variety of computer platforms an enable larger datasets to be collects, leading to better statistics on the resulting velocity fields.

Keywords- PIV; GPU

I. INTRODUCTION

Particle image velocimetry (PIV) has become a powerful tool for studying aerodynamic flow. However, the time required to process PIV data can be considerably large which limits the maximum size of a dataset, leading to concerns about statistical convergence. Normally, PIV data is processed on serial computers, utilizing multiple cores and multithreading technology to speed up calculations. Despite advances in serial computing hardware, the processing remains too slow for large datasets. Therefore, PIV algorithms are being developed to run on parallel architectures equipped with graphics processing units (GPUs).

Most PIV algorithms use cross-correlation-based methods which are extremely parallelizable and therefore can be considerably accelerated using GPUs. Executing a PIV window deformation algorithm with multiple GPUs on a 256×256 pixel image, researchers were able to increase processing speed by 120 times, resulting in a processing speed of 30 pairs per second [1]. In another study, by storing matrices as texture maps on the GPU, images were processed at a speed of 13 pairs per second for a 1024×1024 pixel image [2]. Taking a different approach, researchers using a gradient-based cross-correlation algorithm (FOLKI) on an NVIDIA Tesla C1060 GPU achieved a $50\times$ speedup on 1376×1040 pixel images, resulting in a processing speed of 5 image pairs per second [3]. This is an immense speedup, considering that analyzing a single image pair on serial computers can take on the order of one minute.

It is clear that GPUs can be used in various PIV algorithms to accelerate data processing. Some commercial PIV software has added support for GPUs, but the code is proprietary and thus cannot be developed, optimized for specific architectures, or ported to different architectures such as ARM processors or supercomputing clusters. Therefore, the open-source PIV data processing code OpenPIV has been developed to utilize GPUs. Transferring the cross-correlation portion of the algorithm to the GPU resulted in a $10\times$ speedup when compared to running on a CPU. The total processing time only decreased by a factor of 2. The majority of processing time was spent on the CPU performing sub-pixel peak approximations and velocity field calculations, both of which could be accelerated using a GPU. Another shortcoming of OpenPIV is it does not have full support for multi-pass. A PIV multi-pass algorithm first calculates a coarse velocity field (usually on a 64×64 pixel grid), then uses that information to calculate a finer velocity field. This process repeats until noise levels becomes too great and the highest resolution velocity field is achieved. Furthermore, for complex flows with turbulence and regions of high shear, the absence of a multi-pass PIV algorithm results in more spurious vectors in the final velocity field. Correcting for the spurious vectors is computationally expensive and can not be ported to a GPU, inevitably slowing down the computations. Developing the multi-pass portion of OpenPIV to execute on the GPU would greatly reduce the number of spurious vectors as well as the processing time.

Having a fast, open-source platform for PIV data processing using GPUs would be a valuable tool for multiple researchers. The code can run on any platform, and can be optimized for specific systems. For this to be fully realized, more of the code must be rewritten to run on the GPU, and full support for multi-pass must be added. To ensure universality of the code across

platforms, the OpenPIV software will be tested on a desktop computer with a GPU, an embedded CPU-GPU development system, and the SOSCIP GPU-accelerated supercomputing cluster at SCINET.

II. Methods

All software development is done using an NVIDIA Jetson TX1, referred to from now on as the TX1, which is an embedded CPU-GPU platform for developing GPU accelerated software. The TX1 includes a quad-core ARM Coretx-A57 processor (host) with 4GB of RAM, a 256-core NVIDIA Maxwell GPU, and a variety of I/O ports for peripherals. The host runs the algorithm and transfers certain calculations to the GPU. The data transfer between the host and the GPU is time consuming and often comprises most of the GPU related processing time. Therefore, maximal speed is obtained by strategically and efficiently managing data transfer between the host and GPU. For further testing, the software will be run on a desktop CPU with 16GB of RAM and an NVIDIA GEFORE GT 710 GPU, as well as the SOSCIP cluster at SCINET which includes 14 IMB servers, each with 2×10 core CPUs with 512GB of RAM and 4 NVIDIA Telsa P100 GPUs.

The version of OpenPIV being developed was originally written in Python. However, Python is unable to execute on the GPU. To run code on an NVIDIA GPU, the CUDA interface must be used. CUDA is a layer of software that allows certain languages, such as C, C++, and Fortran, to execute on the GPU. Therefore, the library PyCUDA [4], a Python wrapper for CUDA functions, was used to insert C code into OpenPIV and interface with the GPU. Additionally, some functions from the GPU accelerated Scikits-CUDA library were used. For increased speed, OpenPIV was compiled using Cython.

The PIV algorithm works by cropping an image pair into smaller overlapping regions called interrogation windows (IW) and then uses a FFT-based cross-correlation algorithm on each IW pair to calculate the velocity field. A depiction of the process is shown in Figure 1. This portion of the algorithm has already been written for the GPU, resulting in a $10 \times - 35\times$ speedup depending on IW properties. To further improve OpenPIV, multi-pass capabilities will be added to the software.

A PIV multi-pass algorithm iteratively refines the resolution of the velocity field until the maximum resolution is achieved. First, a coarse velocity field is calculated over the domain. Then, the grid is refined and corrections to the velocity on the finer grid are calculated. This process is then repeated until the noise becomes excessive, or the properties in each grid cell are not optimal for PIV calculations. The iterative method being developed in OpenPIV is based on the work done by Scarano and Riethmuller [5]. The multi-pass algorithm in OpenPIV must be parallelized to run on the GPU. The algorithm contains multiple loops for bounds checking, correction calculations, interpolations, and velocity calculations that can be massively accelerated using the GPU.

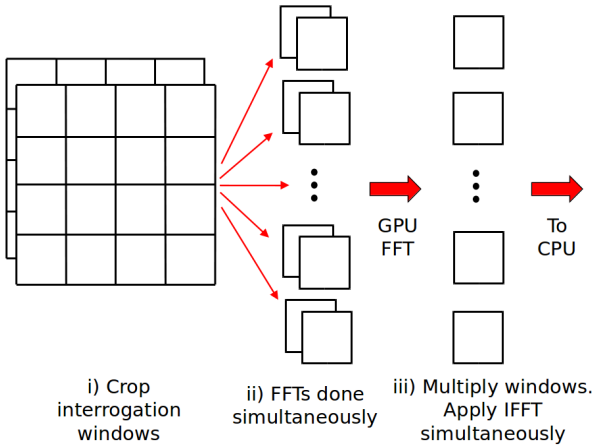

Figure 1. Scheme for calculating FFTs on the GPU

III. Expected Results

The OpenPIV software will be tested on two datasets. The first contains data on a flow control experiment of low Reynolds Number airflow over an airfoil. The experiment was done at a range of angles of attack using a variety of flow control parameters. Details of the experimental setup can be found in [6]. 1000 image pairs were taken at a resolution of 2560 x 2160 pixels. This data was originally processed using the commercial software package LaVision, which provides a set of results for comparison.

The second dataset is open-source and was originally intended to test PIV uncertainty quantification algorithms [7]. The experiment consists of a single turbulent jet with multiple sets of images taken at different regions of the jet, including the jet-core and shear layer. The dataset at each location contains an accurate velocity field calculated from ultra-high resolution images as well as a set of lower resolution images to use with PIV software.

For both datasets, three properties will be investigated. First, the difference in processing time between the GPU-accelerated and CPU version of OpenPIV will be calculated. It is expected that the GPU-accelerated version will be more than $2\times$ faster than the CPU version as more portions from the previously accelerated algorithm are being moved to the GPU. Second, the number of spurious vectors produced in the final velocity field with and without multi-pass will be compared. It is expected that the multi-pass version will produce far less spurious vectors as it should reduce the relative noise level in the calculations. Third, velocity field results of the full GPU-accelerated, multi-pass algorithm will be compared to the LaVision and high resolution velocity field in each dataset. Comparisons for each image pair, the resulting velocity field, and other statistics will be made. When previously compared to LaVision, the CPU version of OpenPIV which did not use multi-pass performed well, except in regions of high shear, such as the boundary layer over the airfoil. The addition of multi-pass is expected to greatly improve

the accuracy of the algorithm, especially in regions of turbulence and high-shear.

The processing time will be tested on three different computer platforms to investigate how the algorithm performs across architectures. The platforms, mentioned before, are the Jetson TX1, a desktop computer equipped with an NVIDIA GPU, and the SOSCIP GPU-accelerated supercomputing cluster at SCINET. The comparison between the TX1 and the desktop computer will be interesting, as the TX1 has a two times more powerful GPU, but the desktop has four times more RAM. It is still expected that the TX1 will outperform the desktop computer, as the bulk of the calculations will be done on the GPU.

Despite the fact that the SCOSCIP cluster is expected to massively outperform the other platforms, the results can be used to infer a maximum PIV dataset size when using supercomputing clusters. When using normal desktop computers, reasonable PIV datasets range from 500 - 1000 images. This puts severe limits on the statistical convergence of some calculations, and increasing the image size by an order of magnitude or more using a supercomputer cluster would be extremely advantageous.

IV. Conclusion

Running PIV algorithms on GPUs can greatly decrease processing time. The open-source PIV data processing software OpenPIV is currently being developed to run on a GPU with added support for multi-pass. The software is written in a combination of Python and CUDA C which is then compiled using Cython, which can run on any architecture that has a CUDA capable GPU. Once complete, OpenPIV will be tested in a variety of datasets for speed and accuracy. The software will also be run on an embedded computer, desktop computer, and supercomputing cluster to compare performance. Ultimately, this project will provide an open-source tool for PIV analysis that will operate on a wide variety of computing platforms.

References

[1] S. Tarashima, S. Someya, and K. Okamoto, "Acceleration of Recursive Cross-Correlation PIV Using Multiple GPUs," Proceedings of the ASME/JSME 2011 8th Thermal Engineering Joint Conference, 2011.

[2] T. Schiwietz and R. Westermann, "GPU-PIV," Vision, Modeling and Visualization, 2004, pp. 151–158.

[3] F. Champagnat, A. Plyer, G. Le Besnerais, B. Leclaire, S. Davoust, and Y. Le Sant, "Fast and accurate PIV computation using highly parallel iterative correlation maximization," Experiments in Fluids, vol. 50 (4), 2011, pp. 1169–1182.

[4] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, and A. Fasih, "PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation," Parallel Computing, vol. 38 (3), 2012, pp. 157–174.

[5] F. Scarano and M. L. Riethmuller, "Iterative multigrid approach in PIV image processing with discrete window offset," Experiments in Fluids, vol. 26 (6), 1999, pp. 513–523.

[6] M. Feero, S. D. Goodfellow, P. Lavoie, and P. E. Sullivan, "Flow Reattachment Using Synthetic Jet Actuation on a Low-Reynolds-Number Airfoil," AIAA Journal, vol. 53 (7), 2014, pp. 1–10.

[7] A. Sciacchitano, D. R. Neal, B. L. Smith, S. O. Warner, P. P. Vlachos, B. Wieneke, and F. Scarano, "Collaborative framework for PIV uncertainty quantification: the experimental database," Measurement Science and Technology, vol. 26 (7), 2015, p. 074004.