# Parallel-Computing Architecture for JWST Wavefront-Sensing Algorithms

**J. Scott Smith, Bruce H. Dean, Alexander Rilee, Thomas P. Zielinski**
*Optics Branch / Code 551*
*NASA Goddard Space Flight Center*
*Greenbelt, MD 20771*

## ABSTRACT

The James Webb Space Telescope (JWST) is the successor to the Hubble Space Telescope and will be NASA's premier observatory of the next decade. Image-based wavefront sensing (phase retrieval) is the primary method for ground testing and on-orbit commissioning. For ground tests at NASA's Goddard Space Flight Center (GSFC) and Johnson Space Center (JSC), near-real-time analysis is critical for ensuring that pass/fail criteria are met before completion of a specific test. To address this need we have developed a computational architecture for image processing and phase retrieval. Using commercially available off-the-shelf hardware and software, we have designed, implemented, and tested a solution for high-speed parallel computing. The architecture is a hybrid solution utilizing both CPUs and GPUs and exploiting the unique advantages of each. Discussions are presented of the architecture, performance, and current limitations.

Keywords: phase retrieval, wavefront sensing, James Webb Space Telescope (JWST), Graphics Processing Units (GPU), Central Processing Unit (CPU)

## 1. INTRODUCTION

The James Webb Space Telescope (JWST) is the next-generation successor to the Hubble Space Telescope. It is a large, space-based infrared telescope scheduled for launch in 2014. From the L2 Lagrange point, JWST will peer through dusty clouds to see stars forming planetary systems. JWST's infrared capabilities will allow astronomers to investigate how galaxies were formed a few hundred million years after the Big Bang; the light from the youngest galaxies and stars are seen in the infrared due to the expansion of the universe.

JWST consists of an Optical Telescope Element (OTE) that sends light to the Integrated Science Instrument Module (ISIM). The OTE is a three-mirror anastigmat, with a 6.5-meter-diameter primary mirror comprised of 18 individual hexagonal segments. The ISIM consists of 5 science instruments (SIs) that make measurements with different portions of the JWST field of view. The 5 SIs are the Near InfraRed Camera (NIRCam), the Near InfraRed Spectrograph (NIRSpec), the Mid-InfraRed Instrument (MIRI), the Fine Guidance Sensor (FGS), and the Fine Guidance Sensor Tunable Filter (TFI) Camera. [1,2,3]

Image-based wavefront sensing (phase retrieval) is the primary method for verifying the SIs, system-level ground testing and integration, and for on-orbit commissioning. [4,5] Each SI, when applicable, will be tested using through focus measurements. [6, 7] Furthermore, phase retrieval will be a critical piece of ground calibration for the integration of each SI into ISIM, when tested at GSFC, [8] and then later when ISIM is integrated with the OTE and spacecraft at JSC. [9] Finally, phase retrieval will be used on-orbit in multiple steps of the Wavefront Sensing and Control Commissioning process and Wavefront Monitoring & Maintenance. [10,11] During each stage of the project, it becomes increasingly more important to achieve near real-time performance of the phase-retrieval results due to the increasing cost and complexity of each test.

## 2. ALGORITHM OVERVIEW

Phase retrieval is an image-based wavefront-sensing approach that recovers the wavefront of the light exiting an optical system. Phase retrieval uses measurements of the irradiance images measured at the detector plane and when possible a model prediction or measurement of the irradiance profile in exit pupil. There are numerous approaches to the phase-retrieval problem [12], but for this paper we will focus on an iterative-transform Misell-Gerchberg-Saxton (MGS) like algorithm. [13] Furthermore, the core of the iterative-transform algorithm adds an adaptive

Fourier kernel using the Hybrid Diversity Algorithm (HDA). HDA extends the dynamic range of a traditional MGS and addresses non-common path terms associated with each image. Additional details of HDA are found elsewhere. [14] In addition to HDA, we are using the Variable Sampling Mapping (VSM) algorithm to address the stability and convergence of an iterative-transform phase-retrieval algorithm for challenging scenarios, such as under-sampled, broadband, known extended-objects, or near in-focus images. VSM was developed to address these cases specifically that arose in the JWST SIs other than NIRCam. Additional details and the verification of VSM are found elsewhere. [15]

In addition to recovery of the exit pupil wavefront, GSFC has developed an architecture that integrates our suite of algorithms for recovering various optical parameters. This architecture is referred to as VYBRANT, and a basic block diagram is shown in Fig. 1. Details of VYBRANT will be discussed at a later date, but briefly it is comprised of four algorithms; a plate scale or the image sampling recovery algorithm [16], a phase-retrieval algorithm as discussed above as HDA-VSM or elsewhere [12], a jitter or extended object recovery algorithm [17], and a pupil amplitude recovery algorithm. [6] The approach in VYBRANT builds upon a bootstrapping method as shown by example in earlier work. [18] Furthermore, VYBRANT is structured with well defined and generic interfaces between each block in Fig. 1, which allows each block to be replaced with an alternate algorithm during convergence. This allows the bootstrapping process to increase the fidelity of a model or algorithm during the recovery, which can then be fine-tuned to optimize run-time performance.

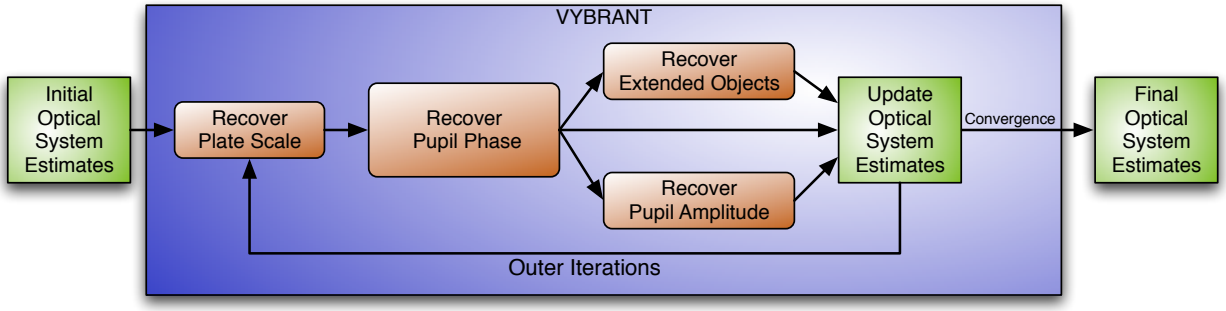



Fig. 1: Simplified functional block diagram for VYBRANT

Independent of the exact algorithm chosen for each step, for most cases related to JWST, majority of the execution time for VYBRANT is in the recover pupil phase block in Fig. 1. This is true even for situations where the fidelity of the model or algorithm complexity is increasing during the recovery process. The pupil phase recovery step requires several orders of magnitude more calls to a two-dimensional fast Fourier Transform (2-D FFT) algorithm then the other blocks. The majority of the research in this paper outlines how to improve the performance for a 2-D FFT while continuing to maintain high-speed interfaces between the various components of VYBRANT. This in turn, will improve the over-all performance of VYBRANT to address the needs for JWST.

## 3. COMPUTATIONAL ARCHITECTURAL OVERVIEW

Prior to the development of VYBRANT and VSM, GSFC developed a high-speed performance system of our phase-retrieval algorithms using high-end Digital Signal Processors (DSPs). [19] The DSPs architecture achieved a performance increase of 730x over current state-of-the-art CPUs in a similar footprint and power requirement. The biggest drawback for the DSPs was the turn-around time between algorithm development in MATLAB and code transported to the DSPs. Furthermore, the DSP have commercially-off-the-shelf (COTS) libraries for the 2-D FFT but they are constrained to powers-of-two matrix sizes. For phase retrieval on data with narrow-band illumination or when making the narrow-band illumination assumption on polychromatic data, it is possible to repose the phase-retrieval problem such that a power-of-two 2-D FFT can be utilized. Yet for broadband light, when using VSM it is non-trivial to constrain the problem to only use power-of-two sized matrices for the 2-D FFT. [15,20]

To address the need for a high-speed performance of VYBRANT there were several goals building upon the lessons learned from earlier work:

1. Minimize the turn-around time from algorithm development to a high-speed implementation.
2. Perform high-speed phase retrieval for a single realization or data set.

3. Analyze multiple phase retrievals for measured data sets simultaneously or simulate realizations for integrated modeling exercises or faster monte-carlo experiments.
4. Allow the flexibility for multiple users, with dedicated hardware for analysis.

Although goals 2 and 3 are closely related, there is a distinct difference. One can think of this as bandwidth versus latency. Goal 2 is to minimize the latency for a single realization, or the time between phase retrievals on individual data sets. Goal 3 is to have a high-bandwidth solution for many realizations including both real and simulated data.

To address these goals, we've developed an eight-node homogenous cluster consisting of COTS components. Each node consists of a quad-core hyper-threaded CPU (8 threads per CPU), an NVIDIA GTX GPU, the maximum amount of RAM available at the time of purchase, and a small hard-drive for the operating system and software. In Fig. 2, we show the construction of the cluster. The head-node is a Mac Pro, and provides an interface to the outside world and multiple users, as shown in Fig. 3.

(a) (b) (c)

Fig. 2: Overview of the construction of the computational cluster; (a) building one of the nodes, (b) installing the software onto one of the nodes, (c) the fully functional cluster.

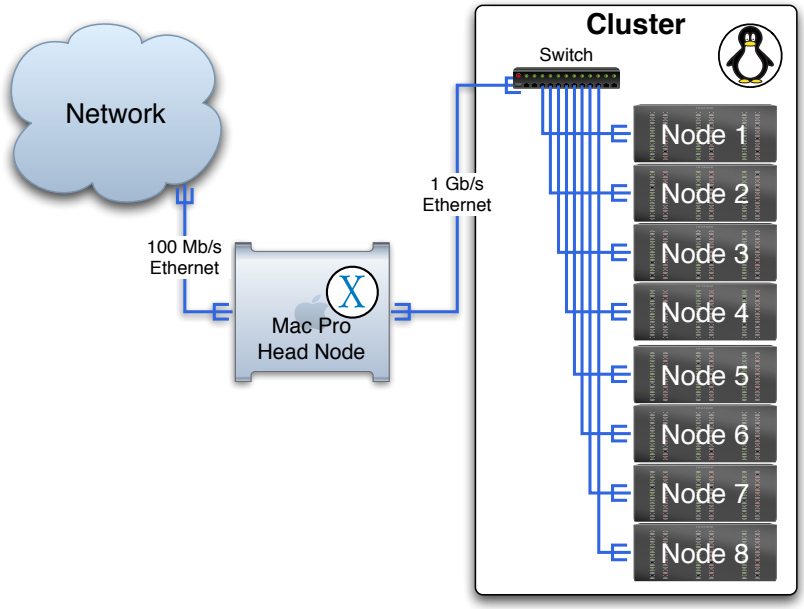
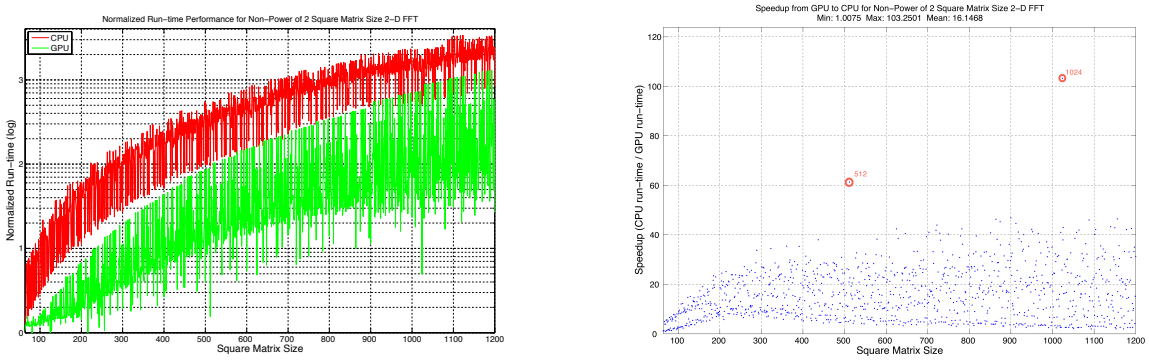



Fig. 3: Architecture of the computational cluster.

# 4. CURRENT PERFORMANCE

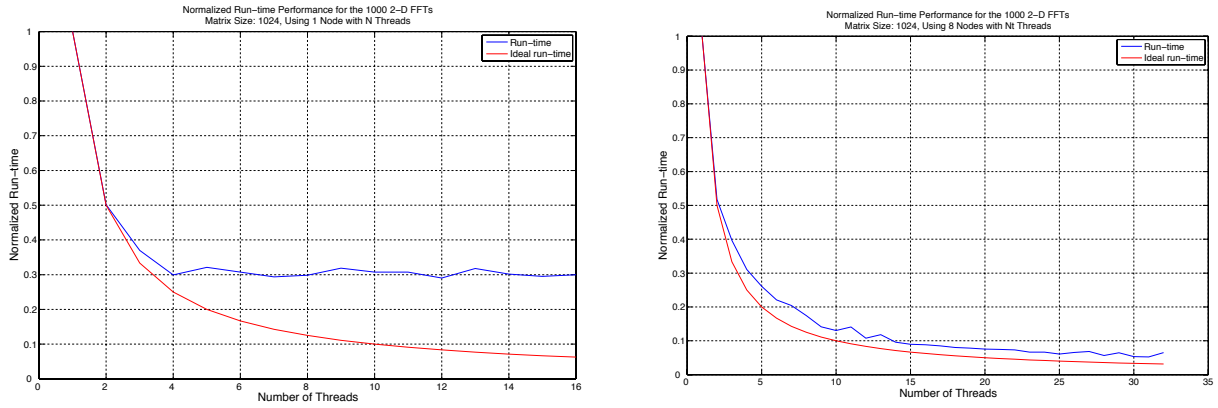
To address the needs outlined in section 3, we chose to focus on COTS hardware and software components. One of the fastest growing industries in high-speed scientific computing is the use of the GPU, [21] and our findings support the motivation of using the GPU as a scientific computational engine. To address the need for a faster 2-D FFT, we present results for run-time performance of the CPU and the GPU in Fig. 4(a). For both systems, we are using MATLAB [22], which uses for the CPU, the FFTW, the fastest non-platform specific FFT engine that is publicly available. [23] For the GPU, we are using Jacket, an AccelerEyes product [24] which specifically helps address the first goal above by providing a MATLAB interface to the GPU. We focus on the non-power-of-two cases to address the broadband illumination problem. This is of interest for several reasons. As shown in Fig. 4(b), the GPU is not always significantly faster than the CPU, but on average is 16.1x faster. This is because the run-time performance of the 2-D FFT is related to the balance of two things; the computational steps of performing the 1-D FFT on each row and column, and the matrix transpose. The tiered memory architecture found in a modern CPU or GPU uses data and instruction locality to increase performance. The matrix transpose performs optimally for numbers with large prime factors, which decreases the number of cache-misses, while the computational step performs optimally for numbers with small prime factors, which decreases computational complexity. It is the balance of these two constraints which causes the CPU-to-GPU speed-up to have a large variance for all matrix sizes. In practice, for broadband illumination problems we have found that hand-tuning the right matrix sizes can account for a speedup of 86x faster. This comes from hand-picking the best CPU and GPU times over the given range of matrix sizes, while adequately sampling the spectrum (or all matrix sizes). The fastest speedup occurs for the 1024 matrix size, which is 103x faster.




(a) (b)

Fig. 4: A comparison of the 2-D FFT for various square matrix sizes, (a) normalized run-time performance for a CPU(red) and GPU (green), and (b) the speed-up from the CPU to the GPU.

For Fig. 4, we looked at the performance of a single CPU with one thread, and a single GPU. Now, we will consider the problem where the CPU can execute multiple threads, thus dividing the total number of 2-D FFTs among the threads. Recall, that we have a quad-core CPU with hyper-threading, which means in theory 8 threads can be executed in parallel. In Fig. 5(a), we show the normalized run-time performance for executing a fixed number of 1000 2-D FFTs on a single CPU, while increasing the number of $N_t$ threads. Additionally, we plot the ideal run-time performance, which is based on the ideal $1/N_t$ curve. The actual run-time and ideal run-time track closely until the number of threads is 4, at which point adding more threads provides negligible speed improvement. This is somewhat expected, although one might anticipate an additional improvement up to 8 threads. Additionally, in Fig. 5(b), we show the normalized run-time performance for using 4 threads per node, for all 8 nodes in the cluster. This produces results very close to the ideal run-time, because it is optimally distributed.




(a) (b)

Fig. 5: Normalized run-time performance for the cluster using $N_t$ threads to perform 1000 2-D FFTs on a 1024 square matrix: (a) using a single physical node, and $N_t$ threads, and (b) using all 8 nodes with 4 threads per node.

For the final results, we present the performance for using VYRBRANT running on a single workstation, versus running on the computational cluster using the GPUs and CPUs. We will consider two cases that are applicable for the JWST testing requirements: (1) a near ideal case, where the parameters of the optical system under test are well calibrated and known at the start of the test with-in their error bars at the start of the test, and (2) a pathological case, where the unknowns in the optical system exceed their predicted error bars. The first case is applicable for the quick-look assessment during a JWST test. The second case is applicable for diagnosing problems seen during the test or commissioning. Furthermore, each case when running on the cluster can be broken down into sub-cases: (a) where the intermediate results are displayed on the screen during the retrieval, and (b) where the cluster does not display intermediate results, and focuses purely on achieving the final answer. The sub-cases (a) and (b) are a trade between speed and useful diagnostics.

For case (1), we consider a simulation where the plate scale, pupil amplitude, and extended object are known to a tolerable level. Thus, VYBRANT is calling the recover pupil phase only. In this case, the speedup from the workstation to the computational cluster is 33.7 for (a) and 54.7 for (b). In case (2), VYBRANT is using the bootstrapping process to increase the knowledge of the various optical parameters. In the scenarios where plate-scale changes inside of VYBRANT, it is difficult to anticipate the matrix size for the 2-D FFT. This can cause some performance degradation. For this case, the speedup is 12.8 for (a) 13.6 for (b). In both cases, the computational cluster and the workstation results agree to with-in the numerical precision of the machine.

| | Workstation Only | Cluster (CPU & GPU) display intermediate results (a) | Cluster (CPU & GPU) do not display intermediate results (b) |
|---|---|---|---|
| (1) Idea case, with all optical parameters within tolerances | 52.9 | 1.57 | 0.96 |
| (2) Pathological case, with optical parameters outside of tolerances | 100 | 7.84 | 7.33 |

Table 1: Normalized run-time for VYBRANT: normalized such that case (2) on the workstation is 100 units of time.

## 5. CONCLUSIONS

In conclusion, we have shown a significant performance increase using the computational cluster to meet the timing demands for JWST. Of interest to users outside of JWST, we explored the performance increase when using a GPU for the 2-D FFT when using non-power-of-two matrix size. Furthermore, we showed that using COTS libraries and hardware can provide a speedup of ~100x for the 2-D FFT, and a speedup of ~55x in practice for a complicated algorithm.

The authors would like to thank the GSFC Wavefront Sensing and Control group for their support in helping to procure and construct the computational cluster. We also thank the JWST OTE, MSE, and ISIM teams for their support and feedback.

## 6. REFERENCES

[1] http://www.jwst.nasa.gov

[2] P. A. Sabelhaus, D. Campbell, M. Clampin, J. Decker, M. Greenhouse, A. Johns, M. Menzel, R. Smith, and P. Sullivan, "An overview of the James Webb Space Telescope (JWST) project," Proc. SPIE, Vol. 5899, (2005).

[3] Clampin, M., "Overview of the James Webb Space Telescope Observatory", Frontier Science Opportunities with the James Webb Space Telescope, (2011).

[4] Feinberg, L. D., Dean, B. H., Aronstein, D. L., Bowers, C. W., Hayden, W., Lyon, R. G., Shiri, R. Smith, J. S., Acton, D. S., Carey, L., Contos, A., Sabatke, E., Schwenker, J., Shields, D., Towell, T., Shi, F., and Meza, L. "TRL-6 for JWST Wavefront Sensing and Control". Proceedings of SPIE (2007) vol. 6687 (08).

[5] Greenhouse, M. A., Drury, M. P., Dunn, J. L., Glazer, S. D., Greville, E., Henegar, G., Johnson, E. L., McCloskey, J. C., Ohl, R. G., Rashford, R. A., and Voyton, M. F. "Status of The James Webb Space Telescope Integrated Science Instrument Module System", Proceedings of SPIE (2010) vol. 7731 (08).

[6] Dean, B. H., Zielinksi T. P., Smith J. S., Bolcar B. R., Aronstein D. L., Fienup J. R., "Phase and Pupil Amplitude Recovery for JWST Space- Optics Control", Frontiers in Optics – OSA (2010).

[7] Smith, J. S., Aronstein D. L., Davila P., Dean B. H., Dorner B., Gnata X., Melf M., Pittet J., and Te Plate M. B. "Optical wavefront characterization using phase retrieval for the NIRSpec demonstration model for the James Webb Space Telescope", Proceedings of SPIE (2010) vol. 7731 (03).

[8] Davila, P. S., Bos, B. J., Eichorn, W. L., Frey B. J., Greeley, B. W., Hakun, C. F., Kubalak, D. A., Leviton, D., Pham, T., Cheng, E. S., Garza, M., Kirk, J., Robinson, D. F., Gong, Q., Guzek J., Hovmand, L., Nagle, A., Nyquist, R., Sabatke, D., Sullivan, J. F., Volmer, P., VonHandorf, R., and Youngworth, R. N., "The Optical Telescope Element Simulator for the James Webb Space Telescope", Proceedings of SPIE (2008) vol. 7010 (3F).

[9] Atkinson, C., Matthews, G., Waldman, M., Wertheirmer, A., Whitman, T., and Oschmann, J., "Architecting a revised optical test approach for JWST", Proceedings of SPIE (2008) vol. 7010 (0Q).

[10] Acton, D. S., Towel, T., Schwenker, J., Shields, D., Sabatke, E., Contos, A. R., Hansen, K., Shi, F., Dean, B. H., and Smith, J. S., "End-to-end commissioning demonstration of the James Webb Space Telescope", Proceedings of SPIE (2007) vol. 6687 (06)

[11] Lightsey, P. A., Chaney, D., Gallagher B., Brown, B., Smith, K., Lewis, J., Barto, A., Knight, J. S., Acton, D. S., Stewart, C., and Siegel, N. "Optical performance for the actively controlled James Webb Space Telescope", Proceedings of SPIE (2010) vol. 7731 (0B).

[12] Fienup, J. R., "Phase retrieval algorithms a comparison", Applied Optics (1982) vol. 21 (15) pp. 2758-2769.

Fienup, J. R., "Phase-retrieval algorithms for a complicated optical system", Applied Optics (1993) vol. 32 (10) pp. 1737-1746.

[13] Gerchberg, R. W., and Saxton, W. O., "Phase Determination from Image and Diffraction Plane Pictures in an Electron Microscope," OPTIK 34, 275 (1971).

Gerchberg, R. W., and Saxton, W. O., "A Practical Algorithm for the Determination of Phase from Image and Diffraction Plane Pictures," OPTIK 35, 237-246 (1972).

Saxton, W. O., “Computer Techniques for Image Processing in Electron Microscopy,” Advances in Electronics and Electron Physics, Supplement 10, L. Marton and C. Marton, eds. (Academic Press, New York, NY, 1978).

[14] Dean, B. H., Aronstein, D. L., Smith, J. S., Shiri, R., and Acton, D. S., "Phase retrieval algorithm for JWST flight and testbed telescope," Proc. SPIE (2006) vol. 6265 (11).

[15] Smith, J. S., Aronstein, D. L., Dean, B. H., and Acton, D. S., “Phase Retrieval on Broadband and Under-sampled Images for the JWST Testbed Telescope,” Proceedings SPIE (2009) vol. 7546.

[16] Aronstein, D. L., and Smith, J. S., “Recovery of the Image-Plane Sampling Parameter Q within Iterative-Transform Phase Retrieval”, Frontiers in Optics – OSA (2011).

[17] Smith, J. S., “Iterative Transform Phase Diversity: An object and wavefront recovery algorithm”, Frontiers in Optics – OSA (2011).

[18] J. R. Fienup, J. C. Marron, T. J. Schulz, and J. H. Seldin, “Hubble Space Telescope characterized by using phase retrieval algorithms,” Appl. Opt. 32, 1747-1767 (1993).

Dean, B. H., “Looking at Hubble through the Eyes of JWST”, IEEE Aerospace Conference, ISBN: 1-4244-0525-4 (2007).

[19] Smith, J. S., Dean, B. H., Haghani, S., “Distributed computing architecture for image-based wavefront sensing and 2D FFTs”, Proceedings of the SPIE (2006) vol. 6274 (21).

[20] Fienup, J., “Phase retrieval for undersampled Broadband Images”, Journal of the Optical Society of America A (1999) vol. 16 (7) pp. 1831-1837.

[21] Owens, J. D., Houston, M., Luebeke, D., Green, S., Stone, J. E., and Phillips, J. C., “GPU computing”, Proceedings of the IEEE (2008) vol. 96 (5) pp. 879-899.

[22] MATLAB® is a registered trademark of The MathWorks, Inc.

[23] Frigo, M., and Johnson, S. G., “FFTW: An Adaptive Software Architecture for the FFT”, http://www.fftw.org.

[24] “The Jacket Platform: Productivity platform for GPU Computing”, (2010), http://www.accerlereyes.com.