Simulating the Performance of Pyramid Wavefront Sensors on Extended Objects and Broadband Sensing

Etsuko Mieda[a], Jeffrey Fung[b, c], Jean-Pierre Veran[a]

[a]National Research Council Herzberg Astronomy and Astrophysics, 5071 W Saanich Rd., Victoria, BC, Canada;
[b]University of California, Berkeley, Campbell Hall, #501, Berkeley, CA, USA;
[c]Sagan fellow

ABSTRACT

We modify the pyramid wavefront sensor simulation code PASSATA to enable simulations for wavefront sensing with extended objects and in broadband. Our modifications make use of Graphics Processing Unit (GPU) with parallel algorithms. We compare the computational time among a variety of CPUs and GPUs, and find that our new GPU implementation improves the Fast Fourier Transfer speed by a factor of 150, and as a result, the entire wavefront reconstruction simulation speed by a factor of 20. As an example for science applications, we study the optimal optical gain, a measurement of pyramid wavefront sensors sensitivity, for different guide object sizes.

Keywords: Pyramid wavefront sensor, Wavefront sensing simulation, Thirty Meter Telescope

1. INTRODUCTION

Pyramid wavefront sensors (PWFSs) offer improved sensitivity and flexibility compared to traditional WFSs, and current and future major facilities will make use of them for better adaptive optics (AO) performance. To optimize design and accurately project performance, quantitative simulations of PWFS are crucial; however, it is computationally expensive to accurately simulate the pupil image resulting from the modulation of the spot around the apex of the pyramid. As telescopes become bigger, and AO requirements become tighter (e.g., high contrast imaging), the dimensions of such simulations quickly become overwhelming for normal computers. To speed up calculations, simulation codes, such as PASSATA and MAOS, have implemented routines that take advantage of accelerated Fast Fourier Transformation (FFT) calculations using Graphics Processing Unit (GPU) massively parallel algorithms. We generalize their approaches by multiplexing the calculations for several off-axis rays and several wavelengths.

We briefly describe how PWFS works in §2. In §3, we describe the code modification in detail. The computational speed comparisons among different machines and PUs is reported in §4.1, and some PWFS simulation results are reported in §4.2. In §5, as an example of science application, we investigate the optimal optical gain (G_{opt}) with different guide object sizes. Finally, we summarize the study in §6.

2. PYRAMID WAVEFRONT SENSOR

We refer the reader to the literature[1–3] for detailed descriptions of PWFS. Here we briefly summarize how it works. A PWFS consists of a pyramid shape prism at a focal plane and relay optics to locate a pupil plane on a detector. See, for example, this reference[2] for a PWFS layout. The four faces of the pyramid steer the beam into four different directions and form four separated pupils on a detector. The derivatives of a wavefront can be obtained from the intensity pattern differences in four pupil images.

Most PWFSs also have a fast steering mirror (FSM) before the pyramid prism to include a modulation capability. A modulation refers to an application of tip and tilt (TT) to the incoming beam by FSM to make a

Send correspondence to Etsuko Mieda (Etsuko.Mieda@nrc-cnrc.gc.ca) or Jean-Pierre Veran (Jean-Pierre.Veran@nrc-cnrc.gc.ca).

circle with the focused beam around the apex of the pyramid. No modulation gives the highest sensitivity (i.e., sensitive to dim stars), and larger modulation gives higher dynamical range (i.e., sensitive to bigger aberration). This convenient trade-off between sensitivity and dynamic range by FSM provides flexibility for a diverse range of observation targets and conditions.

3. CODE DESCRIPTION

We modified and generalized the PWFS simulation code PASSATA to enable the wavefront sensing simulation on extended object and in broadband. PASSATA is written in IDL, developed at Arcetri, Italy. The code takes the FFT of a given turbulence phase to calculate the Point Spread Function (PSF), applies a focal plane mask with a pyramid prescription, and takes the inverse FFT to calculate pupil images created by PWFS. Because the FFT calculations are computationally intensive, an alternative GPU module written in CUDA (a parallel computing platform) for this portion of the code is included in PASSATA. When modulation is applied, the positions of PSF at the tip of the pyramid are calculated as an additional off-axis TT, and added to the incoming phase. The pupil images from each TT are averaged to get the final pupil image. Currently we calculate one position per λ/D along a modulation circle. After the pupil image is obtained, it calculates the x- and y-signals, and obtains coefficients for a set of orthognal bases (such as the KarhunenLoève functions) by multiplying the signals to the inverted interaction matrix. This portion of the code remains in IDL and runs on CPUs.

Expanding upon the capability of the GPU module, we added the functionality to simulate extended objects by generalizing the modulation with TT position described above. An input 2D image (e.g., an image of Titan in Figure 4) is resampled to a λ/D resolution, and each pixel position is calculated as a TT. Many pupil images from different TT positions are calculated and summed using their associated intensities as weights. When modulation is applied, the modulated image is calculated first, by shifting and summing up, and is used as a new input image.

For broadband sensing, pupil images at different discrete sets of wavelengths are calculated and averaged. For this experiment, we assume a flat spectrum with no variation in intensity, but different weights may be assigned according to the spectrum of the guide object and atmospheric transparency. When a modulation is applied, the modulation radius m, which is in a unit of λ/D, is defined at the average wavelength ($< \lambda >$). This modulation radius is converted in a physical angle unit and is applied by FSM. Because the modulation radius is defined in a unit of λ/D, as a result, the actual amount of modulation (m_λ) are different for different wavelengths:

$$m_\lambda = m_{<\lambda>} \frac{\lambda}{< \lambda >}. \tag{1}$$

To take advantage of the massively parallel computing capability of GPUs, we perform FFTs in batches using the CUDA FFT library. For extended object simulations, often the total number of FFTs to perform (corresponding to the number of TT) exceeds the total memory available in the GPU, and so memory recycling is required. By efficiently managing memory, we are able to optimize the code for maximal speed gain.

4. RESULTS

In this section, we use simple simulation setups to compare the computational time among different PUs and demonstrate their accuracy. The setup consists of a non-evolving turbulence and a non-evolving G_{opt}. All noise flags are set to off to exclude any noise in the simulation. We tested three different machines for our study: Macbook Pro with an Intel i7-48470HQ CPU, UCB desktop with an Intel i5-6500 CPU and GTX980Ti GPU, and NRC server machine with a Xeon E5-2670 CPU and a GTX1080Ti GPU.

4.1 Computational time comparison

We ran the same simulation code with and without the GPU module on different machines and measured the computational time. The simulation parameters for this test is summarized in Table 1 Test 1, and the PUs we tested are listed in Table 2. The IDL-only simulations run exclusively on CPUs, and the GPU-module-enabled simulations run partially on GPUs. Since the GPU-module-enabled simulations still run partially on CPUs, we list the two separate computational times, t_{FFT} and t_i, in Table 2. t_{FFT} is the time the simulation takes to

Table 1. Test Parameters

Test	D[a] [pixel]	f[b] [Hz]	N_{sub}[c]	s_{CCD}[d] [pixel]	m[e] [λ/D]	G_{loop}[f]	Δ_f[g]	wind speed [m/s]	G_{OG}[h]
1	512	1000	60	180	5	1	0	0	NA
2	600	800	60	180	5	0.52	2	8.6	0.02

[a]Simulated telescope diameter in pixel. [b] Sampling frequency in Hz. [c] Number of subapertures in telescope diameter. [d] CCD size in pixel. [e] PWFS modulation radius in a unit of λ/D. [f] Closing loop gain. [g] Frame delay in close loop. [h] G_{opt} track gain.

Table 2. Test results

Simulation	Guide Object[a]	λ[b] [nm]	Machine	CPU	GPU	t_{FFT}[c] [s]	t_i[c] [s]
A	Point	750	Mac	Intel i7-4870HQ	···	14.9	15.6
B	Point	750	UCB	Intel i5-6500	···	14.7	15.0
C	Point	750	NRC	Xeon E5-2670	···	17.1	18.1
D	Point	750	UCB	Intel i5-6500	GTX980Ti	0.1	0.5
E	Point	750	NRC	Xeon E5-2670	GTX1080Ti	0.1	1.0
F	Point	BB	UCB	Intel i5-6500	GTX980Ti	0.9	1.4
G	Point	BB	NRC	Xeon E5-2670	GTX1080Ti	0.6	1.9
H	Extended	750	UCB	Intel i5-6500	GTX980Ti	46.5	46.8
I	Extended	750	NRC	Xeon E5-2670	GTX1080Ti	33.8	34.8

[a]"Point" refers to a point source modulated for $5\lambda/D$ with 32 points along a circle. "Extended" is an uniformly illuminated disk of r=0.41 arcsec (the size of Titan at the closest position to Earth) modulated also for $5\lambda/D$, which corresponds to 22,252 points. [b]Wavelength of wavefront sensing in nm. BB for broadband wavefront sensing between 730 and 770 nm at $\Delta\lambda = 10$ nm. [c]t_{FFT} for time spent on FFT and t_i for time spent on whole calculation in one iteration in seconds.

compute the pupil image and PSF on the CPU (if the GPU module is turned off) or the GPU; while t_i is the time it takes for the entire wavefront reconstruction calculation in one iteration, including t_{FFT}.

First, to confirm that the simulation results are independent of the machines, PUs, and/or simulation languages, we compared the root-mean-square (RMS) evolution of the residual wavefront errors for a point guide star at a single wavelength of 750 nm: Simulation A through E in Table 2. Figure 1 shows the RMS relative difference for all machines and PUs with respect to Simulation A (left) and difference between CPU and GPU (right). Interestingly, on the same machine but between CPU and GPU, the relative difference is within the round-off error in single precision numbers ($\sim 10^{-6}$); however, between different machines, the relative differences are small enough ($\sim 10^{-5}$ to 10^{-4}) to be considered negligible but systematically distinctive in comparison to round-off errors, which may imply differences in software (i.e. different versions of IDL) or even hardware can play a role.

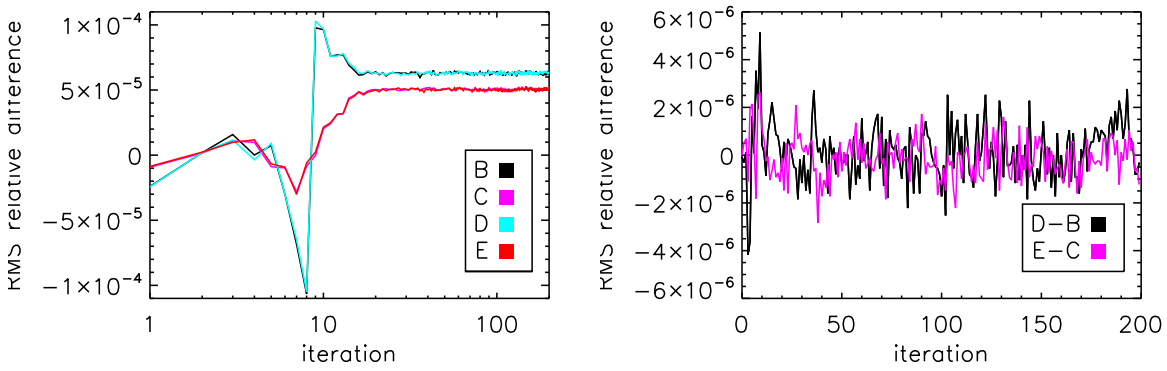

Figure 1 Left: RMS relative differences for different machines and PUs with respect to Intel i7-4870HQ (Simulation A in Table 2). Each color corresponds to the relative difference for Simulation B (black), C (magenta), D (cyan), and E (red). Simulations performed on the same machine are nearly identical; B and D nearly overlap each other and so do C and E. Right: RMS relative difference for the same machine but different PUs. UCB machine (black) and NRC machine (magenta).

The time comparison shows that t_{FFT} is about 150 times or more faster when the GPU module is enabled (Table 2). Because each iteration spends about 0.5 to 1 second on common IDL procedures on CPU, the benefit of the GPU module is particularly pronounced when more data points are calculated, such as the cases with extended objects and/or in broadband.

4.2 Wavefront Reconstruction

In the previous section, we reported the computational time comparison among different machines and improvement enabled by the GPU module. In this section, we report the simulation results and show how wavefront sensing performance changes with sensing wavelengths and guide objects. All results in this section were done using the NRC machine with the GPU module enabled.

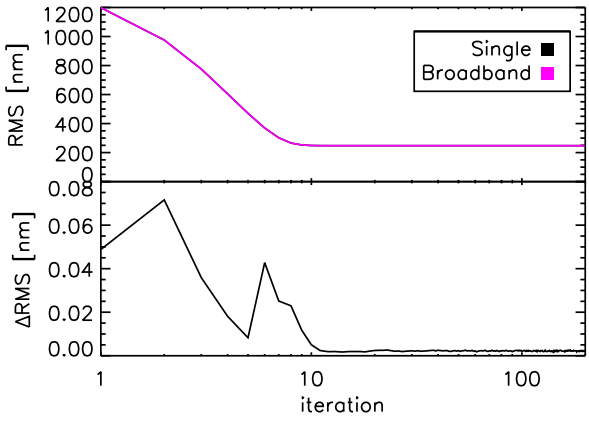

Figure 2 Top: RMS evolution of the residual wavefront errors for a point source as a guide star at a single wavelength of 750 nm (black) and in broadband (magenta). The broadband sensing is done at five different wavelengths: 730, 740, 750, 760, and 770 nm. The two lines nearly overlap one another. Bottom: The RMS difference between broadband sensing and single wavelength sensing. The final RMS difference is 0.002 nm.

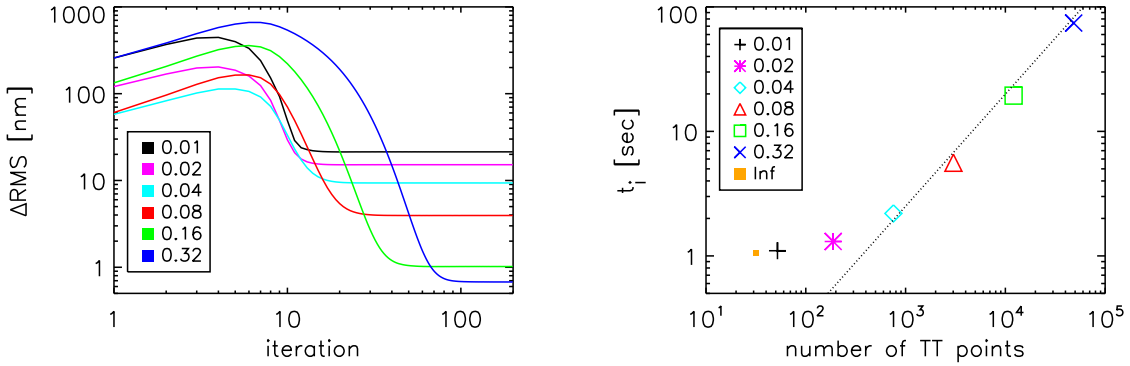

Figure 3 RMS differences between extended objects without modulation and a point source with $m = 5$ modulation (left); and computational time per iteration for the different extended object sizes (right). Uniformly illuminated disks with r = 0.01 (black plus), 0.02 (magenta star), 0.04 (cyan diamond), 0.08 (red triangle), 0.16 (green square), and 0.32 (blue x) arcsec are used. All simulations are measured at a single wavelength $\lambda = 750$ nm. For reference, dotted black line on the right traces linear scaling.

The top panel of Figure 2 shows the RMS evolution of the residual wavefront errors for a point guide star at a single wavelength of 750 nm (black) and in broadband ([730, 740, 750, 760, and 770] nm, magenta). Since the two lines are indistinguishable from this plot, the bottom panel shows the difference between broadband and single-wavelength. Because our broadband is only 40 nm wide for this test, the residual is not too different, but indeed higher for broadband by 0.002 nm.

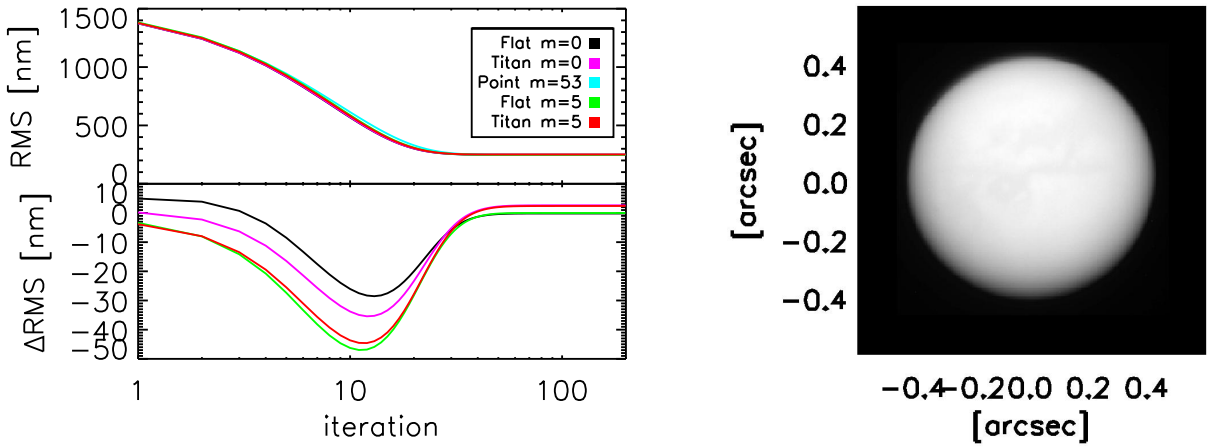

Figure 4 Left: The evolution of RMS for a r=0.41 arcsec uniformly illuminated disk with (green) and without (black) a modulation of 5 λ/D, Titan with (red) and without (magenta) a modulation of 5 λ/D, and a point source (cyan) with a modulation of 53 λ/D, which is the average radius weighted by the uniform disk's intensity. To show the difference more clearly, the bottom panel shows the RMS differences for the extended cases in reference to the point source case. Right: Cassini Titan image, scaled to r=0.41 arcsec (Titan at the closest to Earth).

Figure 3 shows the RMS evolution of a wavefront residual error when flat disks with different sizes are used as guide objects. In this test, the extended objects are not modulated. To show the differences clearly, the RMS evolution of a point source with $m = 5$ modulation is subtracted from all extended results. Because the intensity distribution is uniform and symmetric, the final performance is almost the same ($\leq$ 20 nm). The figure shows that generally the bigger the guide object is, the more iterations it takes for RMS to converge, which is expected because this setup uses a constant optical gain. We also observe a lower final RMS when the guide object is larger, converging back to the modulated point source model. This is probably because our simulation samples the image to λ/D per pixel, and smaller objects are less circular, and asymmetric. This will be tested in the future by changing the sampling scaling.

On the right panel of Figure 3 we see how computational time scales with the number of TT points. As alluded to in Table 2, when we have only a few TT points the code speed is CPU-limited and is around one sec per iteration. As the number of FFT to perform increases with TT points, the code becomes GPU-limited at around 10^3 TT points.

Figure 4 on the left shows the comparison of wavefront sensing performance between the flat disk and the Cassini Titan image (right), both scaled to r=0.41 arcsec in size. Again, to show the differences clearly, the bottom panel shows the RMS differences for extended objects using a point source with $m = 53$ modulation as reference. $m = 53$ modulation corresponds to the average radius of a r=0.41 arcsec uniform disk when the intensity is used as a weight. Because Titan is pretty close to but not exactly a uniformly illuminated perfect disk, the final RMS is higher for the real Titan image but only by a small amount. Because in this test, we did not include any noise and evolving turbulence, we do not see the trade-off between dynamical range and sensitivity due to the modulation.

5. APPLICATIONS

Using the sped up code, we can study the change of PWFS behavior for a range of parameters. In this section, we present preliminary results of ongoing investigations. The simulation parameters for these examples are shown in Table 1 Test 2. To be realistic, the turbulence is evolving, and G_{opt} is iteratively calculated and applied. The noise flags are kept off for this experiment. All examples here are done using the UCB machine with the GPU module enabled.

We study how the optimal optical gain scales with guide object sizes. We expect that the bigger the guide object is, the lower the PWFS sensitivity becomes, and thus G_{opt} has to be smaller to compensate for it. Knowing this scaling will allow us to use the optimal G_{opt} from the beginning to reduce the time the AO loop takes to settle in real on-sky observations.

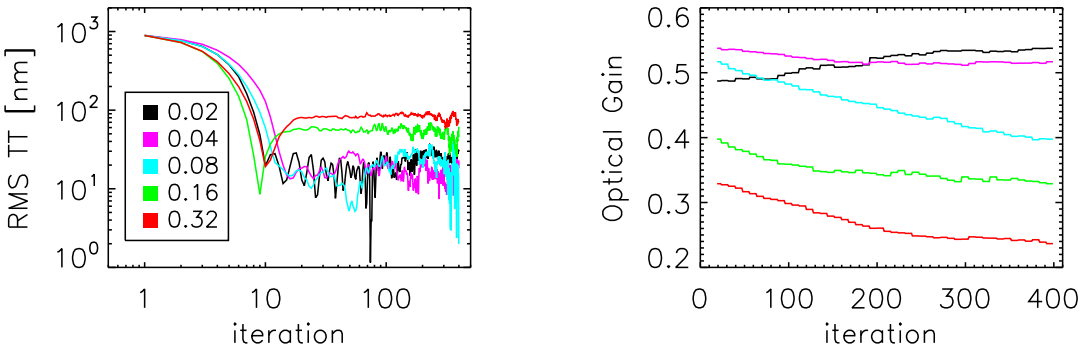

Figure 5 RMS TT (left) and evolution of optical gain (right) for the first 400 iterations for different guide object sizes: r=0.02 (black), 0.04 (magenta), 0.08 (cyan), 0.16 (green), and 0.32 (red) arcsec.

Figure 5 shows the evolution of optical gain for five different extended objects. Tentatively, we find that G_{opt} reduces from 0.55 to 0.25 as the guide object size increases from r=0.02 to 0.32 arcsec. In the left panel of Figure 5, we find fluctuations that not only persist after 400 iteration, but even appear to be growing in amplitude. Whether this implies numerical divergence or is simply a coincidental isolated event demands further investigations.

6. CONCLUSION

We have demonstrated that our GPU module is about 150 times faster than its CPU counterpart, which enables us to perform simulations on extended guide objects and broadband sensing at speeds on the order of seconds per iteration (§4.1). The GPU module maintains the same precision and accuracy as the CPU code (Figure 1) and behaves correctly under non-evolving turbulence (Figures 2, 3, and 4). In the more realistic simulations with time-dependent turbulence, we recover the expected trends of a higher RMS and lower optical gain when the guide object size is larger (Figure 5).

Development for this code will continue in two directions. First, further speedup can be gained from rewriting more portions of the code into CUDA to run on GPUs. This will remove the CPU bottleneck when the number of TT points is below 1000 (Figure 3; right panel), and potentially gain another order of magnitude in speed. Second, our broadband sensing implementation is currently only for a point source, and broadband sensing on an extended object will demands more realism. Namely, for different wavelengths the input image need to be resampled using different λ/D, and the corresponding m_λ should be applied.

While in this work we have used Titan as an example, applications of wavefront sensing using extended guide objects are virtually endless: binary stellar systems, stellar clusters, and even galaxies. Broadband sensing is equally important as it can greatly increase photon count. As PWFS become more prevalent, numerical simulation codes such as ours will become increasing productive in telescope and instrument design. This code will be used locally at NRC-Herzberg, but we also encourage to community to reach out to us when the need arises.

ACKNOWLEDGMENTS

The authors would like to thank Malcolm Smith at NRC-Herzberg for help building the GPU machine at NRC. This work was partly performed under contract with the Jet Propulsion Laboratory (JPL) funded by NASA through the Sagan Fellowship Program executed by the NASA Exoplanet Science Institute.

REFERENCES

1. R. Ragazzoni, "Pupil plane wavefront sensing with an oscillating prism," *Journal of Modern Optics* **43**, pp. 289–293, Feb. 1996.
2. E. Mieda, M. Rosensteiner, M. van Kooten, J.-P. Veran, O. Lardiere, and G. Herriot, "Testing the pyramid truth wavefront sensor for NFIRAOS in the lab," in *Adaptive Optics Systems V*, **9909**, p. 99091J, July 2016.
3. E. Mieda, J.-P. Veran, O. Lardiere, S. Liu, M. Lamb, P. Turri, D. Andersen, and G. Herriot, "Current Status and Implementation of Pyramid Truth Wavefront Sensor on NFIRAOS Simulation Bench at NRC-Herzberg," in *Adaptive Optics for Extremely Large Telescopes 5*, July 2017.