Computer simulations and real-time control of ELT AO systems using graphical processing units

Lianqi Wang, Brent Ellerbroek

Thirty Meter Telescope Project, 1111 S. Arroyo Pkwy, Suite 200, Pasadena, CA, 91105, USA

ABSTRACT

The adaptive optics (AO) simulations at the Thirty Meter Telescope (TMT) have been carried out using the efficient, C based multi-threaded adaptive optics simulator (MAOS, http://github.com/lianqiw/maos). By porting time-critical parts of MAOS to graphical processing units (GPU) using NVIDIA CUDA technology, we achieved a 10 fold speed up for each GTX 580 GPU used compared to a modern quad core CPU. Each time step of full scale end to end simulation for the TMT narrow field infrared AO system (NFIRAOS) takes only 0.11 second in a desktop with two GTX 580s. We also demonstrate that the TMT minimum variance reconstructor can be assembled in matrix vector multiply (MVM) format in 8 seconds with 8 GTX 580 GPUs, meeting the TMT requirement for updating the reconstructor. Analysis show that it is also possible to apply the MVM using 8 GTX 580s within the required latency.

1. INTRODUCTION

The adaptive optics (AO) systems for future ground based extremely large telescopes are so complex that end-to-end numerical simulations in the time domain are an essential part of detailed performance analysis. The AO simulations at TMT for the first light AO system NFIRAOS have been carried out using the C based multi-threaded adaptive optics simulator (MAOS, freely available at http://github.com/lianqiw/maos).[1] It achieves >20 fold speed up compared to the previous MATLAB based linear AO simulator (LAOS), thanks to parallelization and efficient code.

General purpose computing with graphics processing units (GPGPU) has been increasingly used in high performance computing and AO real time controllers (RTC)[2,3]due to unprecedented computing power and improved programmability. To improve simulation efficiency and study the potential application for the NFIRAOS RTC, we ported performance-critical parts of MAOS, such as wavefront sensing, wavefront reconstruction, performance evaluation, etc. to GPU using NVIDIA CUDA technology.[4] We chose the consumer card GTX 580 instead of the Tesla GPU that is dedicated to high performance computing due to its lower cost (1/4 the price) and comparative single precision floating point operation capability. We therefore implemented the above routines in GPU with single precision floating point numbers, which proves to be adequate in most cases.

We achieved about a 10 fold speed up for each GTX 580 used compared to a modern quad core CPU with hyper-threading enabled. Each time step of a full scale end to end NFIRAOS simulation takes 0.1 second in a desktop with two GTX 580s, which means a minute of real time can be simulated in slightly more than an hour. This has made possible simulation work that was not feasible in the past, such as simulation based point spread function reconstruction.[5]

In the mean time, we studied several numerical solvers for the minimum variance reconstructor (MVR)[6,7] planned to be used in the NFIRAOS RTC. We found that 30 iterations of the conjugate gradient (CG) tomography algorithm, from six order 60x60 LGS WFS to six turbulence layers (with four layers oversampled) can be accomplished in 25 milliseconds in a single GTX 580. The Fourier domain preconditioned CG algorithm with 3 iterations can be accomplished in 5 milliseconds, which is only 5 times away from the requirement. However, iterative algorithms are hard to parallelize through multiple GPUs due to bandwidth and latency limitation of the PCI-E interface.

The traditional reconstructor used in RTCs for AO systems is a matrix vector multiply, using a matrix computed by pseudo-inverting the deformable mirror (DM) to wavefront sensor (WFS) gradient influence function

Send correspondence to Lianqi Wang: lianqiw@tmt.org

(referred to as the least square reconstructor, LSR[8–11]). This algorithm is very suitable for RTCs due to straightforward parallelization. However, this MVM is not suitable[6,12] for modern AO systems with multiple laser guide stars (LGS) and/or DMs due to increasing difficulty in inverting the matrix and/or poorer performance without the use of statistical priors. Minimum variance reconstructors with advanced solvers, such as conjugate gradients (CG), Fourier domain preconditioned CG,[13,14] etc have therefore been studied to overcome this obstacle. These advanced solves are usually difficult to implement in a real time RTC due to complexity in parallelization. On the other hand, if we could build the matrix used in MVM using the MVR with advanced solvers, we avoid the two problems associated with the traditional MVM method, yet make the RTC straightforward to implement.

Using an innovative approach, we demonstrated that we can build the matrix for MVM for NFIRAOS in $\sim$80 seconds and/or update it in $\sim$8 seconds using 8 GTX 580 GPUs hosted in a single server [*]. It is also possible to apply the MVM using 8 GTX 580 GPUs in real time within the specified TMT latency of $\leq$1 ms.

The rest of this paper is organized as follows. In Section 2 we give a brief introduction to MAOS. In Section 3 we discussed the porting of MAOS to GPU computing. In Section 4 we describe in detail the implementation of the minimum variance reconstructor in GPU. In Section 5 we describe how we build the matrix for MVM in GPU and discuss how it can be applied in real time. Finally Section 6 gives the conclusion.

2. MULTI-THREADED ADAPTIVE OPTICS SIMULATOR

The Multi-threaded Adaptive Optics Simulator (MAOS) started as a reimplementation of the algorithms in the original, MATLAB®-based linear AO simulator (LAOS). MAOS is written in C language with a function oriented design. MAOS is a complete, end to end AO simulator that is completely configurable through easy-to-read configuration files and command line options, and thus suitable for exploring large parameter spaces. MAOS tries its best to check the configuration for any apparent errors or conflicts. The up-to-date version of documentation and MAOS code can be obtained from `https://github.com/lianqiw/maos/`.MAOS compiles and runs in Linux, Windows (Cygwin), and Mac OS X.

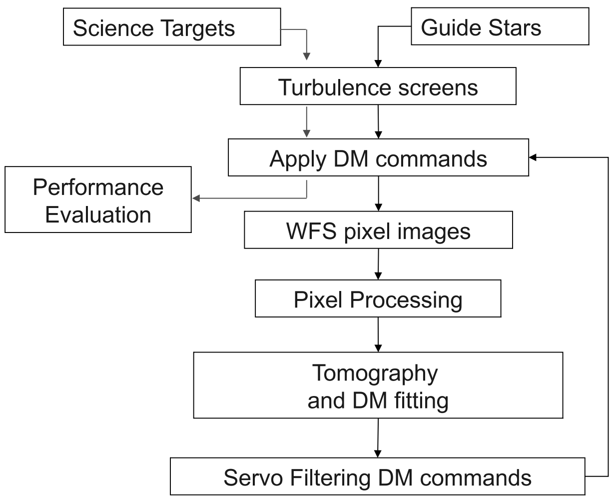

Figure 1. Diagram of AO simulations in MAOS.

Figure 1 shows the block diagram of a basic AO simulations implemented in MAOS. Atmospheric turbulence is represented as one or several discrete optical path difference (OPD) screen(s) at different heights that evolve according to frozen flow with given wind velocity. The resulting aberrations, after corrected by deformable mirror(s) are sensed by one or multiple natural guide star (NGS) or laser guide star (LGS) Shack-Hartmann wavefront sensor(s) (WFS). The sensors can be simulated as idealized wavefront gradient sensors, best Zernike fit tilt sensors, or a physical optics WFS using user specified pixel characteristics and a matched filter[15] pixel processing algorithm. The laser guide star wavefront sensing model includes the impact of guide star elongation

[*] Tyan B7015F72V2R, http://www.tyan.com/product_SKU_spec.aspx?ProductType=BB&pid=412&SKU=600000150

Timing (seconds)	CPU (quad-core)	GPU (1 GTX 580)
Physical Optics WFS	1.36	0.13
Science RMS WFE	0.47	0.05
Tomography (CG30)	0.20	0.025
Tomography (FDPCG3)	0.11	0.007
DM Fitting (CG4)	0.03	0.0025
Total (Phy. WFS)	2.0	0.20

Table 1. Timing on single GTX 580 GPU with 2.8 GHz Intel Core i7 quad-core CPU versus using the CPU alone. The total is less than the sum of components because these steps are running in parallel for better resource utilization.

for a specified sodium layer profile and (optionally) a polar coordinate CCD. The tomographic wavefront reconstruction then estimates the turbulence at one or several different ranges using the pseudo open loop gradients measured by the WFS, using one of several different computationally efficient implementation of a minimum variance reconstruction algorithm as described in.[6] These reconstructed turbulence screens are then fit to the actuators on one or several deformable mirrors (DMs) to achieve the best possible correction over a specified field of view (FoV). Performance evaluation is done in terms of RMS wavefront error, Strehl ratio, and/or point spread function (PSFs) at a few science objects in the target FoV, which might be different from the FoV used for the fitting step. A range of additional specified features are implemented, such as telescope wavefront aberrations and the *ad hoc* and minimum variance "split tomography" control algorithms.[16–18] MAOS can also be configured to evaluate the performance of tomography alone, DM fitting alone, alias-free WFS, etc. For more detailed description, please see.[1]

3. PROGRAMMING IN GPU

We ported mission-critical parts of MAOS, including wavefront sensing, wavefront reconstruction, performance evaluation etc, to run on GPUs using NVIDIA CUDA technology. The selected Nvidia GTX 580 GPU has a total of 512 CUDA cores (or stream processors, equivalent to simplified/dedicated CPU cores) arranged in 16 multiprocessors. It is capable of 1581 GFlops of single precision floating point calculations and 192 GB/s device memory bandwidth. These performance numbers are at least ten times of the state-of-the-art quad core CPUs (i.e., Intel Core i7 2600K). In reality, an average speed up of 10 times is generally observed by porting CPU codes to run on GPUs. Therefore, it is very appealing to run simulations on GPU to improve the simulation speed.

To run codes on GPU with CUDA, we first need to copy data from CPU side memory to GPU memory. Then with "kernel" calls, we can launch many threads that work simultaneously on different parts of the data in user-defined pattern. The finished result can then be copied back to CPU side memory or used for follow up kernels. The "kernel" calls can be scheduled in sequence in streams, and different streams can execute in parallel.

The following sections were ported to GPU computing,

1. Wavefront sensing: including ray tracing from the atmosphere and deformable mirrors to the pupil grid, multiple FFTs per subaperture to get images sampled on detectors, photon and read out noise, gradients calculation from subaperture images, etc.
2. Performance evaluation: including RMS wavefront error computation and point spread function calculation.
3. The wavefront reconstruction: including tomography and deformable mirror fitting for the minimum variance reconstructor.

Significant improvements in speed have been achieved, without losing accuracy due to using single precision floating point numbers. Table 1 shows the timing comparison breakdown of a time step of a TMT NFIRAOS end-to-end simulation. With two GTX 580 GPU and a quad core cpu, the timing per AO time step is reduced further to 0.11 seconds.

Although the performance improvement of $\geq$10 times is significant, we are still far away from the theoretical limits of computing or memory throughput. The main limiting factors are

1. Device memory latency: Each memory operation requires 600 cycles or about 0.3 μs .
2. Kernel launch overhead: Each task is accomplished with a kernel launch which takes 2.3 μs for asynchronous launch and 6.5μs for each synchronization.
3. GPU to CPU interface bandwidth and latency: The current PCI-E $\times$16 interface connecting the graphics card with the mother board has only 8 GB/s throughput and 10μs latency. This single factor makes it almost impossible to use multiple GPUs for tomography using the conjugate gradient algorithm because the synchronization between GPUs slows down the whole process.

4. WAVEFRONT RECONSTRUCTION IN GPU

In this section, we will discuss the implementation of the minimum variance wavefront reconstructor in GPU. The algorithm is based on the formulation in.[6] The wavefront reconstruction can be broken down to two steps, the wavefront tomography and DM fitting.

The tomography step can be summarized by the following equation

$$\begin{aligned} \hat{x} &= (\tilde{H}_x^T G_p^T C_n^{-1} G_p \tilde{H}_x + C_x^{-1})^{-1} \tilde{H}_x^T G_p^T C_n^{-1} g_{ol} && (1) \\ &\equiv R_L^{-1} R_R g_{ol}. && (2) \end{aligned}$$

Here the $\hat{g}_{ol}$ and C_n are the pseudo open loop WFS gradient and its measurement noise covariance, $\tilde{H}_x$ is the ray tracing operator for guide star beam from tomography grid (defined on a few layers and sampled at 1$\times$or $\frac{1}{2}\times$of the subaperture spacing, as shown in Figure 2) to pupil grid and G_p is the influence function from pupil grid to WFS gradients, C_x is the covariance of the atmospheric turbulence, and $\hat{x}$ is the reconstructed wavefront defined on the tomography grid. The right hand side operation $G_x^T C_n^{-1}$is collectively called R_R and the left hand side operation $G_x^T C_n^{-1} G_x + C_x^{-1}$ is collectively called R_L.

The DM fitting step can be summarized by the following equation

$$\begin{aligned} a &= (H_a^T W H_a)^{-1} H_a^T W H_x \hat{x} && (3) \\ &= F_L^{-1} F_R \hat{x}. && (4) \end{aligned}$$

Here, H_x and H_a are ray tracing operators from the tomography grid and DM actuator grid (see Figure 2) to the science focal plane grid along multiple science directions with the weighting defined with W. Again the right and left hand side operations are collectively called F_R and F_L.

These tomography and DM fitting operations are relatively straightforward to implement in GPU due to their intrinsic parallelization. The gradient influence function G_p can be parallelized over subapertures. The non-standard pixel weighting in partially illuminated subapertures need to be passed to GPU for this calculation. The inverse of tomography covariance matrix C_x^{-1} uses the bi-harmonic approximation with periodic boundary conditions so no values need to be passed to GPU. The ray tracing operators $\tilde{H}_x$, H_x, H_a are just interpolations that can easily be parallelized although one need to be careful when the grids have different sampling for maximum efficiency. The transpose ray tracing operators $\tilde{H}_x^T$, and H_a^T need to be implemented as the reverse operation of forward interpolation with atomic operator (fetching memory, accumulating, and storing back done without interruption by other threads) to avoid race condition. The pupil plane weighting matrix W uses gray pixel maps to properly define the pupil, therefore pixel values need to be passed to GPU.

Table 2 shows our measured timing of the operations in tomography. We can see that the achieved GFlops and memory bandwidth are quite below the theoretical throughput, indicating that 1) we are limited by memory throughput, not by Flops, and 2) irregular memory access is severely penalized by device memory operation latency. The inverse ray tracing operation H_x^T is significantly slower than the forward ray tracing operation H_x due to atomic operations used to avoid race condition.

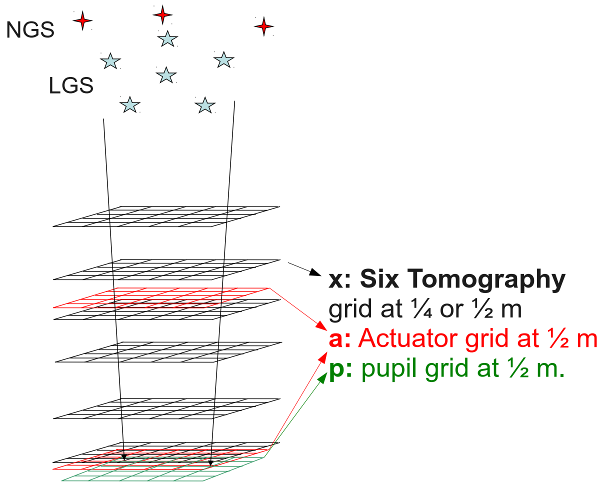

Figure 2. Grid for tomography and DM fitting

	μs	Floating point operations (M Flop)	Throughput (GFlops)	Memory Access (MB)	Bandwidth (GB/s)
H_x	161	4.6	28.4	13.7	83.2
H_x^T	246	4.6	18.6	13.7	54.5
G_p	48	0.56	11.6	1.9	39.0
$C_n^{-1}G_p^T$	91	0.68	22.6	2.1	22.6
C_x^{-1}	85	1.7	19.7	5.5	63.2
R_R	482	5.2	32.1	16	32.1
R_L	603	12	20.0	37	59.9
CG30	23500	382	16.2	1184	49.2
RTC	1000	382	382.0	1184	1184

Table 2. Timing of tomography on a single GTX 580 GPU. The theoretical throughput is 1581 GFlops and 192 GB/s. The row CG30 refers to all the operations for 30 iterations of CG. The row RTC refers to the requirement for RTC with 1 ms latency.

5. MVM IN GPU

As we discussed above, the limited bandwidth ($\sim$10GB/s) and latency (10 μs) between GPU to GPU/CPU communication interface makes it almost impossible to use multiple GPUs for tomography using the iterative algorithms. Therefore we researched another way to overcome this shortcoming and make an RTC in GPU more attractive. We come back to the easy-to-parallelize matrix vector multiply method. Instead of obtaining the reconstruction matrix using the least square reconstructor and/or expensive direct methods such as singular value decomposition (SVD), we can use iterative algorithms, such as CG or FDPCG as described above for the minimum variance reconstructor, to obtain the reconstruction matrix in a more acceptable time with improved performance.

5.1 Obtain the reconstruction matrix

We could obtain the reconstruction matrix by solving the identity matrix

$$E = F_L^{-1} F_R R_L^{-1} R_R I \tag{5}$$

$$\rightarrow (7083 \times 7083)^{-1} \times (7083 \times 62311) \times (62311 \times 62311)^{-1} \times (62311 \times 30984) \times (30984 \times 30984). \tag{6}$$

Here the dimension of each matrix is shown in each parenthesis for the TMT NFIRAOS system. The exponent of -1 indicats that an iterative algorithm instead of a direct inverse of the matrix might be used. We have $n_a = 7083$ (the number of active DM actuators), $n_x = 62311$ (number of points in the tomography grid), and $n_g = 30984$ (the number of LGS WFS gradients. Here split tomography is used where the NGS WFS gradients are used separately to control low order modes only[16]).

Notice that, R_R is a matrix with 30984 columns, meaning there are as many sets as tomography plus DM fitting operations to carry out. However, by solving for the transpose of E instead, we can greatly reduce the number of tomography operations,

$$E^T = R_R^T R_L^{-1} F_R^T F_L^{-1} I \tag{7}$$

$$\rightarrow (30984 \times 62311) \times (62311 \times 62311)^{-1} \times (62311 \times 7083) \times (7083 \times 7083)^{-1} \times (7083 \times 7083).$$

Here we have used the symmetry property of F_L and R_L. In this way, we only need to solve $n_a = 7083$ sets of tomography plus DM fitting operations, which is more than a factor of 4 reduction. Notice that the operations in F_R and R_R are mostly ray tracing and wavefront gradient operations, so the transpose operation is similar to the forward operations. Further notice that the DM fitting parameters do not change when the telescope pupil rotates in NFIRAOS or seeing changes. We can solve it using the Cholesky Back Substitution (CBS) method (an exact solution) and reuse it every time we need to initiate or update the reconstruction matrix.

To obtain the reconstruction matrix the first time, we need to start from a zero guess, so the number of CG or FDPCG iterations needs to be about 10-20 times larger than that required in closed loop simulations that use warm restart.[19] To update the reconstruction matrix when conditions vary (such as pupil rotation or seeing changes), we can simply update from the prior values using warm restart CG or FDPCG, which greatly reduces the number of iterations required.

With precomputed DM fitting results ($F_L^{-1} I$) calculated in CPU using CBS, it takes about 600 seconds to do the remaining operations $R_R^T R_L^{-1} F_T^T$ (see Equation 7) in a single GTX 580 with 50 iterations of FDPCG in tomography. Notice that the n_a columns of $F_L^{-1} I$ are strictly independent and can be solved in parallel in different GPUs. So it will only take about 75 seconds with 8 GPUs. In order to update the reconstruction matrix when conditions vary, we will only need about 5 FDPCG iterations. Therefore it will take about 8 seconds to update the matrix, which meets the TMT requirement for real time reconstructor updates. Without much modification, this same formulation can be used with other variants of MVR, such as the Frim method.[20]

Figure 3 plots shows the performance obtained using MVM with reconstruction matrix computed as described above with FDPCG. We can see that MVM constructed from 50 iterations of FDPCG is about 28 nm in quadrature worse than the baseline CG30 algorithm. With more FDPCG iterations, the performance of MVM should improve further.

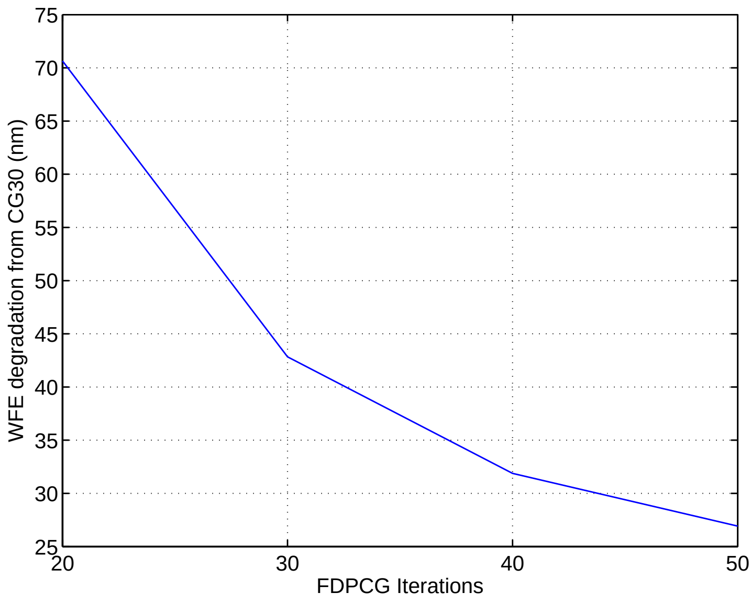

Figure 3. The plot shows the performance using the MVM with a reconstruction matrix computed as described above using FDPCG. The x axis shows the number of FDPCG iterations in R_L^{-1}, and the y axis shows the quadrature difference of the RMS wavefront error with respect to the TMT baseline (30 iterations of warm-restart CG).

5.2 Apply MVM in GPU based RTC

Once we have the reconstruction matrix ready, it is possible to apply it to gradients to compute actuator commands in GPUs with acceptable latency.[2] Suppose each element takes 4 bytes, the storage of the matrix will be $4 \times 7083 \times 30984 = 837$ Mbytes. This fits in the device memory of a single GTX 580 graphics card. But the memory throughput will determine how many GPUs are actually required.

To do the matrix vector multiply, the matrix needs to be brought into GPU from the device memory, and one multiply and accumulation operation (2 floating point operations, or Flops) per matrix element needs to be done. At 1 ms, this requires about 409 giga Flop/s (GFlops) of computing power and 817 giga byte/second (GB/s) of memory throughput. The computing power is within the capacity of a single GTX 580 (1581 GLOPS), but the memory throughput exceeds its capacity of 192 GB/s. With 8 GPUs, the memory throughput requirement of each GPU decreases to about 102 GB/s and should be feasible considering the benign pattern of the memory access.

The next question is whether we can get the gradients into, and actuator commands out of, the GPUs in time. We will partition the problem so that each GPU handle $n_g/8$ number of gradients and accumulate all the actuator command contributions computed from them. The detector read out and gradient computing process is required to complete within 0.5 millisecond after the exposure is done, which is the equivalent of $\sim$620 gradients every 10 microseconds (roughly the total PCI-E interface latency per transfer). We will distributed these gradients evenly to all the GPUs. This is by far below the PCI-E interface bandwidth (8 GB/s) and the latency should not be a problem as long as each transaction of $\sim$ 620 gradients completes within 10 microseconds. Finally the actuator commands accumulated in each GPU should then be copied to the CPU memory (level 2 cache is enough to contain all the data) and summed by the CPU. The memory copy from GPU to CPU can be done in about 13 micro-seconds, and the CPU should be able to sum the commands in about 12 micro-seconds.

Based on considerations above, it should be possible to apply the MVM in 8 GTX 580 GPUs in less than 1 ms. By the time TMT RTC gets built, more powerful GPUs should be out, and the latency requirement might be met by fewer cards.

5.3 Other considerations

5.3.1 Pseudo open loop gradients

For the minimum various reconstructor, the gradient vector on the right hand side consists of pseudo open loop (PSOL) gradients, so the final result needs to be differed with the actuator command that was used to form the PSOL gradients:

$$a = E(g_{cl} + G_a a_0) - a_0. \tag{8}$$

Here a_0 is the DM command vector during WFS integration, and G_a is the DM to WFS gradient operator (sparse matrix of dimension $n_g \times n_a$). This formula can be reformatted as

$$a = Eg_{cl} + (EG_a - I)a_0 \tag{9}$$

where the first part is the same as the ordinary closed loop matrix vector multiply, while the second part is a correction term using the *a prior* information. It can be computed quickly with another MVM (the matrix is only $n_a \times n_a$) before the gradients are available and does not contribute to overall latency.

5.3.2 Minimum variance split tomography

In the minimum variance split tomography formulation, the NGS gradients need to be adjusted by the LGS tomography results

$$\hat{a}_N = R_N(g_N - G_N \hat{x}_L) \tag{10}$$

where g_N is NGS pseudo open loop gradient and G_N is NGS atmosphere to WFS gradient operator. The vector $\hat{x}_L$ is the tomography output of Equation 1. We could precompute the term $R_N^L = R_N G_N R_L^{-1} R_R$ and apply the reconstruction with

$$\hat{a}_N = R_N g_N - R_N^L g_L \tag{11}$$

where R_N^L is only of $n_{g_N} \times n_{g_L}$ with $n_{g_N} \leq 12$.

6. CONCLUSION

In this paper we show that by running performance critical routines in GPUs, we can speed up the adaptive optics simulations by an order of magnitude with consumer graphics cards. Faster simulations makes possible certain studies that require extended time simulation of a real system.

It is also attractive to run the real time controller with GPUs. We demonstrated that we can assemble an end-to-end reconstruction matrix for a minimum variance algorithm to be used in a matrix vector multiply manner that is straightforward to implement in RTC. Analysis show that it should be possible to apply the MVM in real time using 8 GTX 580 GPUs. We are planning to test this next.

Acknowledgment

The TMT Project gratefully acknowledges the support of the TMT collaborating institutions. They are the Association of Canadian Universities for Research in Astronomy (ACURA), the California Institute of Technology, the University of California, the National Astronomical Observatory of Japan, the National Astronomical Observatories of China and their consortium partners, and the Department of Science and Technology of India and their supported institutes. This work was supported as well by the Gordon and Betty Moore Foundation, the Canada Foundation for Innovation, the Ontario Ministry of Research and Innovation, the National Research Council of Canada, the Natural Sciences and Engineering Research Council of Canada, the British Columbia Knowledge Development Fund, the Association of Universities for Research in Astronomy (AURA) and the U.S. National Science Foundation.

REFERENCES

[1] Wang, L. and Ellerbroek, B., "Fast end-to-end multi-conjugate ao simulations using graphical processing units and the maos simulation code," in [*Adaptative Optics for Extremely Large Telescopes II*], (2011).

[2] Bouchez, A. H., Dekany, R. G., Roberts, J. E., Angione, J. R., Baranec, C., Bui, K., Burruss, R. S., Croner, E. E., Guiwits, S. R., Hale, D. D. S., Henning, J. R., Palmer, D., Shelton, J. C., Troy, M., Truong, T. N., Wallace, J. K., and Zolkower, J., "Status of the PALM-3000 high-order adaptive optics system," in [*Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*], **7736** (July 2010).

[3] Basden, A., Geng, D., Myers, R., and Younger, E., "Durham adaptive optics real-time controller," *Applied Optics* **49**(32), 6354–6363 (2010).

[4] CUDA, "http://developer.nvidia.com/what-cuda."

[5] Gilles, L., Wang, L., Ellerbroek, B., Correia, C., and Veran, J.-P., "Point spread function reconstruction for laser guide star tomographyadaptive optics," in [*Adaptive optics for extrame large telescopes II*], (2011).

[6] Ellerbroek, B. L., "Efficient computation of minimum-variance wave-front reconstructors with sparse matrix techniques," *J. Opt. Soc. Am. A* **19**(9), 1803–1816 (2002).

[7] TMT, "NFIRAOS RTC Algorithm," *Available at http://www.tmt.org/business/rtc/* .

[8] Fried, D. L., "Least-square fitting a wave-front distortion estimate to an array of phase-difference measurements," *J. Opt. Soc. Am.* **67**, 370–375 (Mar 1977).

[9] Hudgin, R. H., "Wave-front reconstruction for compensated imaging," *Journal of the Optical Society of America (1917-1983)* **67**, 375–378 (Mar. 1977).

[10] Herrmann, J., "Least-squares wave front errors of minimum norm," *Journal of the Optical Society of America (1917-1983)* **70**, 28 (Jan. 1980).

[11] Wallner, E. P., "Optimal wave-front correction using slope measurements," *J. Opt. Soc. Am.* **73**, 1771–1776 (Dec 1983).

[12] Flicker, R., Rigaut, F. J., and Ellerbroek, B. L., "Comparison of multiconjugate adaptive optics configurations and control algorithms for the Gemini South 8-m telescope," in [*Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*], Wizinowich, P. L., ed., **4007**, 1032–1043 (July 2000).

[13] Vogel, C. R. and Yang, Q., "Fast optimal wavefront reconstruction for multi-conjugate adaptive optics using the Fourier domain preconditioned conjugate gradient algorithm," *Optics Express* **14**, 7487 (2006).

[14] Yang, Q., Vogel, C. R., and Ellerbroek, B. L., "Fourier domain preconditioned conjugate gradient algorithm for atmospheric tomography," *Appl. Opt.* **45**, 5281–5293 (July 2006).

[15] Gilles, L. and Ellerbroek, B., "Shack-hartmann wavefront sensing with elongated sodium laser beacons: centroiding versus matched filtering," *Appl. Opt.* **45**(25), 6568–6576 (2006).

[16] Gilles, L. and Ellerbroek, B. L., "Split atmospheric tomography using laser and natural guide stars," *Journal of the Optical Society of America A* **25**, 2427– (Sept. 2008).

[17] Wang, L., Gilles, L., and Ellerbroek, B., "Analysis of the improvement in sky coverage for multiconjugate adaptive optics systems obtained using minimum variance split tomography," *Appl. Opt.* **50**, 3000–3010 (Jun 2011).

[18] Gilles, L., Wang, L., and Ellerbroek, B., "Minimum variance split tomography for laser guide star adaptive optics," *European Journal of Control, Special Issue on Adaptive Optics* **17**, Nb. 3 (2011).

[19] Lessard, L., West, M., Macmynowski, D., and Lall, S., "Warm-started wavefront reconstruction for adaptive optics," *Journal of the Optical Society of America A* **25**, 1147 (Apr. 2008).

[20] Béchet, C., Tallon, M., and Thiébaut, E., "FRIM: minimum-variance reconstructor with a fractal iterative method," in [*Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*], **6272** (July 2006).