Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **JAN 2006**	2. REPORT TYPE	3. DATES COVERED **00-00-2006 to 00-00-2006**

4. TITLE AND SUBTITLE **Fast Optical Ray Tracing Using Multiple DSPs**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **United States Naval Academy,Annapolis,MD,21402**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Optical ray tracing is a computationally intensive operation that is central both to the design of optical systems and to analyzing their performance once built. The authors have previously reported on the use of parallel digital signal processors (DSPs) to reduce the time required to perform ray tracing in analyzing the performance of the moderate resolution imaging spectroradiometer (MODIS), which is presently in orbit on multiple spacecraft. The earlier work was incomplete, providing only a conservative estimate of the performance improvement that could be achieved with one to four DSPs. This paper reports on the completed project and extends the earlier work to eight DSPs. As predicted in the earlier paper, not all rays make it through the entire optical system. Many are lost along the way. This is one factor that led to reduced processing time. Another is the use of an optimizing compiler. In this paper, the authors present results showing the separate effect of each of these two independent factors on the overall processing time. The most significant finding is the extraordinarily linear relationship between the number of DSPs available and the speed of the ray tracing. By using eight DSPs, the processing time is reduced from two weeks to less than one and a half days, an improvement of nearly a whole order of magnitude. Low-cost high-speed ray tracing is now feasible using off-the-shelf plug-in processor boards.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**9**	

Fast Optical Ray Tracing Using Multiple DSPs

Charles B. Cameron, *Senior Member, IEEE*, Rosa Nívea Rodríguez, *Member, IEEE*, Nathan Padgett, Eugene Waluschka, Semion Kizhner, Gabriel Colón, and Colleen Weeks

***Abstract*—Optical ray tracing is a computationally intensive operation that is central both to the design of optical systems and to analyzing their performance once built. The authors have previously reported on the use of parallel digital signal processors (DSPs) to reduce the time required to perform ray tracing in analyzing the performance of the moderate resolution imaging spectroradiometer (MODIS), which is presently in orbit on multiple spacecraft. The earlier work was incomplete, providing only a conservative estimate of the performance improvement that could be achieved with one to four DSPs. This paper reports on the completed project and extends the earlier work to eight DSPs. As predicted in the earlier paper, not all rays make it through the entire optical system. Many are lost along the way. This is one factor that led to reduced processing time. Another is the use of an optimizing compiler. In this paper, the authors present results showing the separate effect of each of these two independent factors on the overall processing time. The most significant finding is the extraordinarily linear relationship between the number of DSPs available and the speed of the ray tracing. By using eight DSPs, the processing time is reduced from two weeks to less than one and a half days, an improvement of nearly a whole order of magnitude. Low-cost high-speed ray tracing is now feasible using off-the-shelf plug-in processor boards.**

***Index Terms*—Digital signal processor (DSP), moderate resolution imaging spectroradiometer (MODIS), optical ray tracing, optics, parallel processing, reconfigurable computing, resistance–capacitance (RC).**

I. Introduction

RAY-TRACING simulations are an essential part of the design of optical systems as well as analyzing their performance after their construction [1], [2]. Unfortunately, the time required for tracing substantial numbers of rays can be staggering. In an earlier article [3], we reported on preliminary results entailing the use of from one to four digital signal processors (DSP) to perform such simulations for the moderate resolution imaging spectroradiometer (MODIS): an earth-sensing instrument currently in orbit on NASA's Terra and Aqua satellites [4]. We summarized the essentials of optical ray tracing in that article and described the MODIS instrument, whose performance we simulated. In this paper, we explain how we tested the completed system, give measurements of the time taken to complete the simulations, and analyze the relationship between the number of DSPs available and the speed of the system.

A characteristic of the MODIS is that a calibration is performed once in each orbit. Although the MODIS ordinarily takes images of the earth's surface, during calibration it takes an image of the sun (not directly but as projected onto a diffuse surface). This provides a more-or-less uniform illumination of all the picture elements on the image plane's detectors. Because the sun's radiance is so great, its light is attenuated before being projected onto the diffuser. Instead of using neutral-density filters, MODIS uses a screen perforated with numerous pinholes. It reduces the irradiance by 91.5%. Some aspects of the diffuser are discussed in Waluschka *et al.* [5].

After the plane of the entrance aperture, the attenuation screen, and the diffuser, there follow a further 25 optical surfaces starting with a mirror and ending with an image plane containing an array of optical sensors.

II. Previous Results

Generally speaking, only a subset of the optical rays impinging on an optical system ever reaches the image plane. Many rays are lost along the way because they pass outside one or another of the apertures encountered within the optical instrument. In our earlier work [3], we were very conservative in estimating the time required to complete a simulation. We had not then completed the ray-tracing program: It did not yet trace all the rays of interest. We instead repeatedly traced a single ray, one known to reach the image plane, as it proceeded from the diffuser to the image plane.[1]

We measured the time it took to trace 349 932 instances of this single ray and extrapolated from those measurements the time it would take to trace the 6.24×10^9 rays of a complete simulation. These results appear in Table I. A key result is the number of rays traced each second for each parallel processor available: 2848 rays $\cdot$ (s $\cdot$ processor)$^{-1}$. An earlier simulation done using a program written in FORTRAN and executed on a single Digital Equipment Corporation (DEC) Alpha 3000 series model 800 computer achieved a rate of roughly 5160 rays $\cdot$ s^{-1}.

Manuscript received October 18, 2004; revised January 4, 2006. This work was supported in part by the National Aeronautics and Space Administration (NASA) and in part by the United States Naval Academy.

C. B. Cameron is with the United States Naval Academy, Annapolis, MD 21402 USA (e-mail: cameronc@usna.edu).

R. N. Rodríguez and G. Colón are with the Universidad de Puerto Rico, Mayagüez, PR 00681 USA.

N. Padgett is with the Georgia Institute of Technology, Atlanta, GA 30332 USA.

E. Waluschka and S. Kizhner are with National Aeronautics and Space Administration (NASA) Goddard Space Flight Center, Greenbelt, MD 20768 USA.

C. Weeks is with the Johns Hopkins University, Baltimore, MD 21218 USA.

Digital Object Identifier 10.1109/TIM.2006.873813

[1]More recently, once we had completed the program, we confirmed our earlier certainty that many simulated rays never reached the image plane. Every such abandoned ray reduced the overall simulation time, giving apparently better performance than our earlier estimates suggested we should expect.

TABLE I
PREVIOUSLY REPORTED TIME TO TRACE 349932 RAYS

Number of Processors	Elapsed Time / s	Time per Ray /μs	Expected Completion Time / day	Rays Traced / s
1	122.861	351.1	25.4	2 848
2	61.449	175.6	12.7	5 695
3	40.942	117.0	8.4	8 547
4	30.721	87.8	6.3	11 389

Rate of ray tracing: 2 848 rays traced $\cdot$ s^{-1} $\cdot$ processor^{-1}.

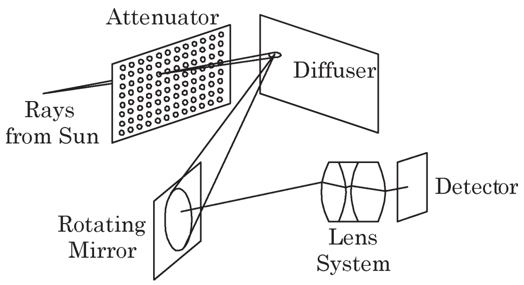
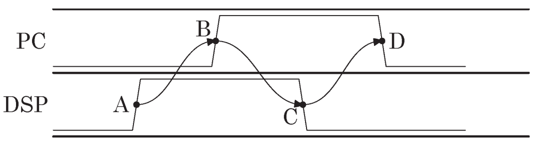


Fig. 1. MODIS optical system.



A: Previous results ready. DSP ready for new data.
B: PC accepts results, assigns new task.
C: DSP waiting for release.
D: PC releases DSP.

Fig. 2. Synchronization of the PC and one DSP. This is a full handshaking scheme using polling. The same arrangement exists for each DSP in the system.

III. EXPERIMENTAL SETUP

A. Implementation

We implemented the system on a Dell Precision WorkStation 530 MT based on an Intel Xeon CPU operating at a frequency of 1.70 GHz. This system used a 400-MHz system bus, an 8-kB L1 cache, and a 256-MB L2 cache. It also employed two Bittware Hammerhead 66-MHz PCI boards, each containing four Analog Devices ADSP21160 DSPs operating at 80 MHz.[2] Our approach was to use the PC to direct and coordinate the operation of the eight DSPs.[3] The PC could have been used to perform some of the ray tracing from the diffuser to the image plane. This was not done, however, because we wanted to make it easy to determine the relationship between the number of DSPs available and the number of rays traced each second. In practice, it would be desirable to let the PC perform ray tracing whenever it was otherwise idle.

B. Simulation

The MODIS optical system is shown schematically in Fig. 1. The attenuator screen has 1951 pinholes. We traced rays from 485 of these, the same ones used in the original FORTRAN simulation program. From the center of each pinhole, we simulated launching a rectangular array of 21 rows $\times$ 21 rays/row $=$ 441 rays. The central ray of this grouping was based on the direction of the sun from the entrance pupil, and the transverse extent of the grouping was based on the sun's angular extent (semidiameter), roughly 0.25°.

We used the host PC to trace each ray from a pinhole to the diffuser. Some of the rays that departed the pinholes missed the diffuser; these were discarded. Of those that struck the diffuser, each ray was then used to generate a second rectangular array of 241 rows $\times$ 121 rays/row $=$ 29 161 rays. This collection of rays was traced by an assigned DSP, which discarded any rays that did not reach the image plane. Of the rays that did reach it, the DSP tallied the number of rays that struck each cell in the image plane. These cells were arranged in 12 rows $\times$ 10 cells/row $=$ 120 cells. Upon completing its assigned task, the DSP notified the PC that it was ready for a new assignment.

When the PC discovered a waiting DSP, it retrieved its 120-element tally and gave it a new point on the diffuser from which to trace a further 29 161 rays. The PC could have accumulated these results. In fact, we chose not to do this, instead saving a subset of the results for inspection and verification. Comparison of numerous 120-element tallies with those from the earlier FORTRAN program showed no difference at all, even though the FORTRAN program used double-precision floating-point arithmetic, whereas the PC and the DSPs only used single-precision floating-point arithmetic.

To synchronize the PC and the DSP, we took advantage of the fact that the Hammerhead board allows two-port access to the memory of the DSP. We implemented a full handshaking scheme, as illustrated in Fig. 2. To do this, we set up a pair of mailboxes in each DSP's memory, one for the DSP and one for the PC. When the DSP was ready for the PC to service it, the DSP set its flag (point A in the figure). The PC polled this flag from time to time and set its own flag once it discovered the DSP had set its flag (B). At this point the DSP reset its flag (C). The PC now retrieved the results of the DSP's latest ray-tracing simulation and gave the DSP the new coordinates in the diffuser plane corresponding to its next assignment. Then, the PC reset its flag, leaving the DSP to start its next task (D).

We performed eight complete simulations. Each simulation entailed tracing all

$$485\text{ pinholes} \times \frac{441\text{ pinhole rays}}{\text{pinhole}} \times \frac{29\,161\text{ diffuser rays}}{\text{pinhole ray}}$$
$$= 6\,237\,100\,485\text{ diffuser rays}$$

and each was performed using a different number of DSPs, from one to eight. The measured simulation times and corresponding rates of tracing rays are shown in Table II. Two factors are responsible for the increase in the ray-tracing rate from that shown in Table I.

1) The completed program was compiled with optimization enabled, leading to a more efficient code in the PC and, more importantly, in each DSP.

[2]In our earlier work [3], we used only a single Hammerhead board.
[3]The programs running in the PC as well as all eight DSPs were written in the C programming language.

TABLE II
RUNNING THE SIMULATION ON DIFFERENT NUMBERS OF PROCESSORS

# of Processors Available	Simulation Time / ks	Rate of Ray Tracing / krays $\cdot$ s^{-1}
1	1 023.877 625	6.091 646 44
2	511.936 109	12.183 357 21
3	341.290 782	18.275 033 53
4	255.967 797	24.366 738 93
5	204.774 219	30.458 426 43
6	170.645 281	36.550 090 62
7	146.267 812	42.641 647 53
8	127.984 234	48.733 350 12

# of rays traced: 6.24×10^9 rays
Rate of ray tracing: 6 092 rays traced $\cdot$ s^{-1} $\cdot$ processor^{-1}

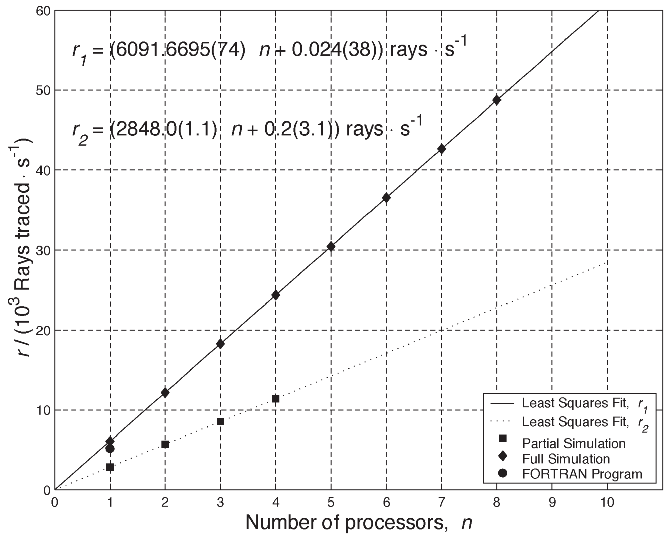


Fig. 3. Comparison of ray-tracing rates.

2) Many rays in the complete simulation never reached the image plane. Being abandoned part way through the entire system, they took less time to trace.

IV. COMPARISON

Fig. 3 provides plots of three different measurements.

1) Rate of ray tracing achieved by the FORTRAN program executing on a DEC Alpha computer. This program could only be executed on a single processor. The rate is marked by a black dot (●).
2) Rates of ray tracing estimated from partial simulations with nonoptimized code. These are from the data presented in Table I. The rates are marked by black squares (■).
3) Rates of ray tracing estimated from full simulations with optimized code. These are from the data presented in Table II. The rates are marked by black diamonds (⧫).

Additionally, the plot shows least square linear fits for the data from the two C programs. Equations for these two lines are

rate of tracing rays under full simulation

$$= ((6\,091.6695 \pm 0.0074)n + 0.024 \pm 0.038)\ \text{rays} \cdot \text{s}^{-1} \quad (1)$$

TABLE III
RESIDUAL ANALYSIS

N	Observed Rate / krays $\cdot$ s^{-1}	Predicted Rate / krays $\cdot$ s^{-1}	Residual / rays $\cdot$ s^{-1}	Residual^2 / rays^2 $\cdot$ s^{-2}
1	6.091 646 43	6.091 693 17	−0.046 743 271	0.002 184 933
2	12.183 357 21	12.183 362 66	−0.005 446 525	0.000 029 665
3	18.275 033 53	18.275 032 14	0.001 395 983	0.000 001 949
4	24.366 738 93	24.366 701 61	0.037 320 391	0.001 392 812
5	30.458 426 43	30.458 371 09	0.055 342 298	0.003 062 770
6	36.550 090 62	36.550 040 57	0.050 053 448	0.002 505 348
7	42.641 647 53	42.641 710 05	−0.062 513 228	0.003 907 904
8	48.733 350 12	48.733 379 53	−0.029 409 097	0.000 864 895
			Sum	0.013 950 275

N = the number of processors available

$$\text{Average residual} = \frac{\sqrt{\text{Sum}}}{8} = 0.014\,763\,91\ \text{rays} \cdot \text{s}^{-1}$$

and

rate of tracing rays under partial simulation

$$= ((2\,848.0 \pm 1.1)n + 0.2 \pm 3.1)\ \text{rays} \cdot \text{s}^{-1} \quad (2)$$

where n is the number of processors assigned.

The stated uncertainties are ± 1 standard deviation. We believe that the uncertainties are smaller in (1) than in (2) because (1) stemmed from tracing 6.24×10^9 rays while (2) stemmed from tracing only 3.5×10^5 rays. It seems unlikely that switching from nonoptimized to optimized compilation would significantly affect the uncertainties.

The fact that the standard deviations associated with the coefficients in these equations are so small shows that there is a very highly linear relationship between the number of processors assigned and the rate at which the work can be performed. In turn, this implies that the administrative overhead associated with managing up to eight processors is negligible. As a result, we should be able to add many more processors to the system before the administrative overhead becomes significant.

Another approach to gauging the suitability of a linear least squares fit is to compare the observed measurements to the values predicted by the fit. One way to do this is to take their differences, square them, add them up, take the square root of this sum, and divide by the number of terms in the sum. Doing this yields 0.01476391 rays $\cdot$ s^{-1}, as shown in Table III. Comparing this to any of the tabulated observations makes it clear that this is a negligible quantity and supports our view that the least squares fit is suitable.

The effect of the master processor (the Dell computer) on the rate of ray tracing has been deliberately ignored. The master processor traces 29 161 rays from the pinhole attenuator to the diffuser. Compared to the 6 237 100 485 rays traced by the DSPs from the diffuser to the image plane, the time taken to trace these early rays is negligible.

V. EFFECTS OF OPTIMIZATION

The graph in Fig. 3 shows a marked improvement in the rate of ray tracing from 2848 rays $\cdot$ (s $\cdot$ processor)$^{-1}$ to

TABLE IV
COMPARISON OF NONOPTIMIZED AND OPTIMIZED VERSIONS OF THE PROGRAM WITH FOUR PROCESSORS AVAILABLE

Program Version	Simulation Time / ks	Rate of Ray Tracing / krays · s^{-1}
Optimized	255.967 797	24.366 738 93
Non-Optimized	439.985 516	14.182 285 04

Speed-up due to optimization = 71.8%

6092 rays · (s · processor)$^{-1}$. As remarked above, this improvement can be attributed in part to the fact that not all rays traverse every optical surface within the instrument, and in part to our having used an optimizing compiler for the faster program.

In order to distinguish between these two effects, we performed a full simulation using four DSPs with a nonoptimized version of the compiled program and compared the time that version took to the time required by the optimized version of the program using the same number of DSPs.

To analyze the results, we define the following quantities.

1) n, as before, is the number of processors assigned.
2) $N = 6\,237\,100\,485$ is the number of rays traced in a full simulation.
3) $N_{\mathrm{P}} = 349\,932$ is the number of rays traced in a partial simulation.
4) $\rho \leq 1$ is the average fraction of all rays that make it to the image plane. Tracing ρN rays to the image plane would take the same time as tracing all N rays and discarding a ray as soon as it is discovered not to reach the image plane. At this point, we still consider it an unknown, but we want to determine its value.
5) r_1 is the rate of tracing those rays that do reach the image plane using the optimized program, and it is measured in rays · (s · processor)$^{-1}$. At this point, we still consider it an unknown, but we want to determine its value.
6) r_2, likewise, is the rate of tracing those rays that do reach the image plane using the nonoptimized program, also measured in rays · (s · processor)$^{-1}$. This is given in (2) as $r_2 = (2848.0 \pm 1.1)$ rays · (s · processor)$^{-1}$, if we ignore the small fixed offset.
7) $\tau_{1,\mathrm{F}} = 255\,967.797$ s $\pm$ 290 μs is the time to execute the full simulation using the optimized program with four assigned processors, assuming the timing measurements are accurate to within ± 0.5 ms over the full duration of the measurement. The clock routines provided in the C runtime library offer the current time to the nearest 1.0 ms. If we assume the round-off error is uniformly distributed in the range [-0.5 ms, 0.5 ms], then the standard deviation of the error is 290 μs [6].[4]
8) $\tau_{2,\mathrm{F}} = 439\,985.516$ s $\pm$ 290 μs is the time to execute the full simulation using the nonoptimized program with four assigned processors, assuming the same clock accuracy (Table IV).
9) $\tau_{2,\mathrm{P}} = 30.721$ s $\pm$ 290 μs is the time to execute the partial simulation using the nonoptimized program with four assigned processors, assuming the same clock accuracy.

Equations (1) and (2) show that the rate of ray tracing is very nearly proportional to the number of processors assigned. Our objective is to find ρ and r_1 given all the other quantities. We can do this by first writing down several relationships:

$$\tau_{1,\mathrm{F}} = \frac{\rho N}{4 r_1} \quad \tau_{2,\mathrm{F}} = \frac{\rho N}{4 r_2} \quad \tau_{2,\mathrm{P}} = \frac{N_{\mathrm{P}}}{4 r_2}. \tag{3}$$

The 4 in each denominator is due to our having used $n = 4$ processors when measuring $\tau_{1,\mathrm{F}}$, $\tau_{2,\mathrm{F}}$, and $\tau_{2,\mathrm{P}}$.

Dividing $\tau_{1,\mathrm{F}}$ by $\tau_{2,\mathrm{F}}$, we have

$$\frac{\tau_{1,\mathrm{F}}}{\tau_{2,\mathrm{F}}} = \frac{\rho N}{4 r_1} \frac{4 r_2}{\rho N} = \frac{r_2}{r_1} \tag{4}$$

therefore

$$r_1 = r_2 \frac{\tau_{2,\mathrm{F}}}{\tau_{1,\mathrm{F}}} = \left((2848.0 \pm 1.1) \text{ rays} \cdot (\text{s} \cdot \text{processor})^{-1}\right) \times \left(\frac{439\,985.516 \text{ s} \pm 290\ \mu\text{s}}{255\,967.797 \text{s} \pm 290\ \mu\text{s}}\right).$$

Optimized ray-tracing rate

$$r_1 = (4895.5 \pm 1.9) \text{ rays} \cdot (\text{s} \cdot \text{processor})^{-1}. \tag{5}$$

This is the rate of ray tracing with an optimized program when every ray is traced all the way to the image plane.[5] A fair comparison with other systems that do ray tracing can be obtained by multiplying this by the number of surfaces traced within the DSPs, which is 25:

ray-tracing rate for fair comparison

$$= (122\,386 \pm 47) \text{ rays} \cdot \text{surfaces} \cdot (\text{s} \cdot \text{processor})^{-1}. \tag{6}$$

In our experiments, then, using eight processors, we achieved

Ray-tracing rate for fair comparison with eight processors

$$= (979\,090 \pm 380) \text{ rays} \cdot \text{surfaces} \cdot \text{s}^{-1}. \tag{7}$$

Now, we turn to the determination of the average fraction ρ of rays that make it to the image plane in our system. Dividing $\tau_{2,\mathrm{F}}$ by $\tau_{2,\mathrm{P}}$, we get

$$\frac{\tau_{2,\mathrm{F}}}{\tau_{2,\mathrm{P}}} = \frac{\rho N}{4 r_2} \frac{4 r_2}{N_{\mathrm{P}}} = \frac{\rho N}{N_P}.$$

[4]We use this method of estimating the standard deviation of the error in all clock measurements throughout this paper. The method neglects any error due to clock drift.

[5]Methods of estimating error propagation are given by Bevington [7].

Solving for ρ, we get

$$\rho = \frac{N_\mathrm{P}}{N}\frac{\tau_{2,\mathrm{F}}}{\tau_{2,\mathrm{P}}} = \left(\frac{349\,932}{6\,237\,100\,485}\right)\left(\frac{439\,985.516 \text{ s} \pm 290\ \mu\text{s}}{30.721 \text{ s} \pm 290\ \mu\text{s}}\right).$$

Effective fraction of rays reaching the image plane

$$\rho = 0.8035334 \pm 0.0000076. \quad (8)$$

Therefore, in our full simulation of the MODIS instrument, we traced N rays, discarding any that went astray; the time taken was roughly equivalent to that needed to trace 80% N rays all the way to the image plane. We can use this result to adjust (1), again ignoring the small fixed offset:

rate of tracing rays under full simulation

$$= (6\,091.669\,5 \pm 0.007\,4) \times (0.8035334 \pm 0.0000076) \text{ rays} \cdot (\text{s} \cdot \text{processor})^{-1} = (4\,894.860 \pm 0.046) \text{ rays} \cdot (\text{s} \cdot \text{processor})^{-1}. \quad (9)$$

Note how consistent this is with the value of r_1 already given in (5). The reason for the difference is that the value in (5) was calculated from $\tau_{1,\mathrm{F}}$ and $\tau_{2,\mathrm{F}}$, whereas the value in (9) was calculated from $\tau_{2,\mathrm{F}}$ and $\tau_{2,\mathrm{P}}$.

We could also calculate ρ in another way using (3) and (5):

$$\rho = \frac{4 r_1 \tau_{1,\mathrm{F}}}{N} = \frac{4\left((4895.5 \pm 1.9) \text{ rays} \cdot \text{s}^{-1}\right)(255\,967.797 \text{ s} \pm 290\ \mu\text{s})}{6\,237\,100\,485 \text{ rays}}$$

$$\rho = 0.80363 \pm 0.00031. \quad (10)$$

Comparing this to (8), we see that both methods yield values of ρ that are very close to one another. The uncertainty from the second method of estimating ρ is larger because it used the shorter execution time associated with a partial simulation, with its correspondingly larger relative error. The close agreement shows persuasively that our simulation was equivalent to tracing just 80% of all the rays we actually traced but following this lesser number of rays all the way to the image plane.

We can calculate the (fractional) speedup in the program due to optimization by using a formula widely used in computer architecture [8]. Here, we compare the tracing rate of the optimized program to that of the nonoptimized program:

$$\text{speedup} = \frac{r_1 - r_2}{r_2} = \frac{(4895.5 \pm 1.9) \text{ rays} \cdot (\text{s} \cdot \text{processor})^{-1}}{(2848.0 \pm 1.1) \text{ rays} \cdot (\text{s} \cdot \text{processor})^{-1}} - 1$$

$$\text{speedup from optimization} = (71.891 \pm 0.094)\%. \quad (11)$$

We can also calculate the apparent speedup due to the fact that, in the full simulation, not all rays reach the image plane. Here, we compare the apparent rate of ray tracing using the optimized program to the rate that would apply if all traced rays indeed reached the image plane:

$$\text{speedup} = \frac{\frac{r_1}{\rho} - r_1}{r_1} = \frac{1}{\rho} - 1 = \frac{1}{0.8035334 \pm 0.0000076} - 1.$$

Apparent speedup due to rays going astray

$$= (24.45033 \pm 0.00034)\%. \quad (12)$$

The speedup given in (11) is real and is due to using an optimized program. That given in (12) is illusory caused by a simulation that traces a large number of rays that do not actually reach the image plane. Quoting this artificial percentage speedup could be highly misleading since it depends on the choice of traced rays and not on the processor speed or the ray-tracing program itself.

The speedup in (11) applies if a single processor is used. We can finally calculate the speedup due to using n processors running an optimized program rather than a single one running a nonoptimized program:

$$\text{speedup} = \frac{n r_1 - r_2}{r_2} = \frac{n r_1}{r_2} - 1 = \frac{n(4895.5 \pm 1.9) \text{ rays} \cdot (\text{s} \cdot \text{processor})^{-1}}{(2848.0 \pm 1.1) \text{ rays} \cdot (\text{s} \cdot \text{processor})^{-1}} - 1$$

speedup using n processors with optimized code

$$= (1.71891 \pm 0.00094)n - 1. \quad (13)$$

Although we used DSPs for our slave processors, these are not ideally suited to the ray-tracing problem because ray tracing requires extensive use of division and square roots.[6] The Analog Devices ADSP21160 DSPs we used do not fully support either of these floating-point operations in hardware. Rather, software is used to perform them with an attendant penalty in performance. It would be far more efficient to use slave processors with hardware that fully supports floating-point arithmetic such as that defined by ANSI/IEEE Std 754-1985, which is the IEEE standard for binary floating-point arithmetic. We used ADSP21160 DSPs for two primary reasons: We had them on hand and they provide a simple interface to a standard host personal computer.

The original FORTRAN program executing on a single DEC Alpha 3000 series model 800 computer took two weeks to perform a full simulation. We reduced this to 35.55 h using two Hammerhead boards with eight ADSP21160 DSPs, which is an improvement of nearly one order of magnitude.

[6]As explained in our earlier paper [3], the need for sine and cosine calculations can be confined to the initialization phase of the program's execution, where the functions that require them are computed exactly once. During the repetitive phase that calculates ray trajectories, trigonometric functions are calculated without using expensive (that is, time consuming) sine or cosine functions.

Comparing the speeds of different processors is often done using the published benchmarks. Getting a comparison between a general-purpose microprocessor such as the DEC Alpha 3000 and a digital signal processor such as the ADSP21160 is difficult because they address different markets and so are evaluated using different benchmarks. A rough comparison can be made by linking together the results of several unrelated benchmarks and treating the processors compared as being fairly represented by all of them. A further assumption in this method is that benchmark scores are linearly related to performance.

Longbottom [9] gives the results of Whetstone benchmarks showing single-precision and double-precision floating-point arithmetic performance of various processors. The Intel Pentium III with a clock frequency of 1.4 GHz received scores of 972 and 1006, with an average of 989. The DEC Alpha series 3000 Model 800 with a clock frequency of 200 MHz received scores of 155 and 129, with an average of 142. A comparison of these two averages permits us to estimate that the Pentium III is 7.0 times faster than the Alpha.

Berkeley Design Technology, Inc. (BDTI) has developed a benchmark for signal-processing speeds [10] with benchmark results published on the Internet [11]. The Analog Devices ADSP-2126x with a clock frequency of 200 MHz received a score of 1090. The Intel Pentium III with a clock frequency of 1.4 GHz received a score of 3130. A comparison of these two scores permits us to estimate that the Pentium III is 2.8 times faster than the 2126x.

Analog Devices publishes comparisons of their 32-bit DSP microprocessors on the Internet [12]. The Analog Devices ADSP-21262 with a clock frequency of 200 MHz can execute up to 1.2×10^9 floating-point instructions per second (1.2 GFLOPS). The Analog Devices ADSP-21160 with a clock frequency of 80 MHz can execute up to 480×10^6 floating-point instructions per second (480 MFLOPS). A comparison of these two scores permits us to estimate that the ADSP-21262 is 2.5 times faster than the ADSP-21160. Combining these factors, we have

$$\begin{aligned}\frac{\text{speed}_{\mathrm{ADSP-21\,160}}}{\text{speed}_{\mathrm{Alpha}}} &= \left(\frac{\text{speed}_{\mathrm{ADSP-21\,160}}}{\text{speed}_{\mathrm{ADSP-21\,262}}}\right) \\ &\quad \times \left(\frac{\text{speed}_{\mathrm{ADSP-21\,262}}}{\text{speed}_{\mathrm{Pentium\ III}}}\right)\left(\frac{\text{speed}_{\mathrm{Pentium\ III}}}{\text{speed}_{\mathrm{Alpha}}}\right) \\ &= \left(\frac{1}{2.5}\right)\left(\frac{1}{2.8}\right)(7.0) \\ &= 0.97 \approx 1.\end{aligned}$$

Such a comparison is crude, at best, and neglects the important fact that Alpha has floating-point hardware to support division [13], whereas the ADSP-21160 does not. Even so, it suggests a roughly comparable level of performance between the two processors. Note, though, that the ADSP-21160 can only achieve the rated speed when processing two sets of data with the same instruction simultaneously. We did not use this mode, so the performance should only be half that of an Alpha 3000.

In fact, the optimized program with one ADSP-21160 assigned took 1023.877625 ks, as shown in Table II, or 11.9 days, compared with roughly 14 days on the Alpha 3000 series model 800. The DSP processor appears to be a little more efficient, therefore, even though it was not running two sets of data simultaneously. This can more likely be ascribed to the differences in the program than in the underlying hardware. A further improvement in the software to take advantage of the DSP's ability to process two sets of data at a time might yield a significant improvement in the ray-tracing rate.

Bittware now offers an improved product, the Danube 6-PaC PCI (D6PC) board with six Analog Devices ADSP-TS201S TigerSHARC DSPs. These operate at 300 MHz, which is a factor of 3.75 higher than that of the Bittware Hammerhead boards we used, operating at just 80 MHz. If we assume that the increased clock rate corresponds to an increased performance rate, then, we can multiply this factor by the value of $122\,388$ rays $\cdot$ surfaces $\cdot$ (s $\cdot$ processor)$^{-1}$ given in (6) to get an estimate of the performance that a system using Danube boards should be able to obtain:

$$\begin{aligned}&3.75 \times 122\,388 \text{ rays} \cdot \text{surfaces} \cdot (\text{s} \cdot \text{processor})^{-1} \\ &\qquad \approx 459\,000 \text{ rays} \cdot \text{surfaces} \cdot (\text{s} \cdot \text{processor})^{-1}.\end{aligned}$$

According to Abbott [14], using a typical PCI bus, we can connect six boards directly to a motherboard. We could therefore envisage using six Danube boards with a total of 36 processors. This would yield

$$\begin{aligned}&36 \text{ processors} \times 459\,000 \text{ rays} \cdot \text{surfaces} \cdot (\text{s} \cdot \text{processor})^{-1} \\ &\qquad = 16.5 \times 10^6 \text{ rays} \cdot \text{surfaces} \cdot \text{s}^{-1}.\end{aligned}$$

We can also consider the effect using six Danube boards would have on the observed apparent ray-tracing rate in (1), using the fastest apparent rate of ray tracing observed in our experiments, as shown in Table II

$$48\,733 \text{ rays} \cdot \text{s}^{-1} \times 3.75 \times \frac{36}{8} = 822 \times 10^3 \text{ rays} \cdot \text{ s}^{-1}$$

and calculate from it the amount of time we could expect it to take to perform a full simulation of the MODIS instrument using six Danube boards with a total of 36 processors:

$$\begin{aligned}\text{Total time} &= \frac{6\,237\,100\,485 \text{ rays}}{822 \times 10^3 \text{ rays} \cdot \text{s}^{-1}} \\ &\approx 2.1 \text{ h}.\end{aligned}$$

The difference would represent a 16.9-fold increase in the rate, corresponding to a speedup of 15.9, or 1590%, and would amount to 2.2 orders of magnitude less than the time required by the original FORTRAN program on a single DEC Alpha computer.

VI. Conclusion

The number r_1 in (5) is a parameter of the ray-tracing system, not of the optical system. It gives the rate of tracing rays through a single surface of an optical system. In our analysis, however, we have ignored the fact that the time to calculate the interaction of light rays with planar surfaces, ellipsoidal surfaces, and nonellipsoidal curved surfaces varies substantially. We have instead tacitly assumed that the mix of such surfaces in the MODIS instrument is representative of most instruments. A more careful analysis would report different values of r_1 for each kind of surface. In comparing two candidate ray-tracing systems, the parameter r_1 is an important one to consider.

The number ρ in (8), on the other hand, is a parameter of the optical system and of the choice of rays used in the simulation, not of the ray-tracing system *per se*. To the extent that ρ is lower than one, the simulations entail tracing some rays part way through the optical system before discarding them. These rays are ill chosen, insofar as it is undesirable to spend any time tracing rays that get discarded. Ideally, we would choose no such rays to simulate.

We could cause ρ to be identically one by, say, tracing a single ray known to traverse the whole system successfully, as we did in our earlier work [3], but then we would be neglecting many rays that give a more complete picture of the performance of the optical system. Simulations could be regarded as most productive when ρ is close to, but still less than, the value one. In comparing two ray-tracing systems, values of ρ should be equal for a fair comparison of any particular simulation.

This paper has made it clear that the ray-tracing problem is very amenable to parallel processing. Since each ray is independent of all others, we can assign any arbitrary subset of rays to available processors for them to trace. An administrative program (executing on the PC in our implementation) can easily gather the results and assign new ray bundles to large numbers of processors without falling behind or unduly delaying the number crunching in the slave processors.

References

[1] M. J. Kidger, *Fundamental Optical Design*. Bellingham, WA: SPIE, 2002.

[2] P. Mouroulis and J. Macdonald, *Geometrical Optics and Optical Design*. New York: Oxford Univ. Press, 1997.

[3] C. B. Cameron, R. N. Rodriguez, N. Padgett, E. Waluschka, and S. Kizhner, "Optical ray tracing using parallel processors," *IEEE Trans. Instrum. Meas.*, vol. 54, no. 1, pp. 87–97, Feb. 2005.

[4] NASA Goddard Space Flight Center. (2004). *MODIS Technical Specifications*. [Online]. Available: http://modis.gsfc.nasa.gov/about/specs.html

[5] E. Waluschka, J. A. Esposito, J.-Q. Sun, X. Wang, and X. Xiong. (2002, Jan.). MODIS solar diffuser: Modeled and actual performance. *Earth Obs. Syst. VI*. vol. 4483(1), pp. 146–155. [Online]. Available: http://link.aip.org/link/?PSI/4483/146/1

[6] G. R. Cooper and C. D. McGillem, *Probabilistic Methods of Signal and System Analysis*, 2nd ed. New York: CBS, 1986.

[7] P. R. Bevington, *Data Reduction and Error Analysis for the Physical Sciences*. New York: McGraw-Hill, 1969.

[8] V. P. Heuring, H. F. Jordan, and M. Murdocca, *Computer Systems Design and Architecture*, 2nd ed. Upper Saddle River, NJ: Pearson, 2004.

[9] R. Longbottom. (2005, Nov.). *Whetstone Benchmark History and Results*. [Online]. Available: http://homepage.virgin.net/roy.longbottom/whetstone.htm

[10] I. B. Berkeley Design Technology, Inc. (2004, Sep.). *The BDTImark2000: A Summary Measure of Signal Processing Speed*. [Online]. Available: http://www.bdti.com/bdtimark/BDTImark2000.pdf

[11] ——. (2005, Dec.). *BDTImark2000 Scores for Floating-Point Packaged Processors*. [Online]. Available: http://www.bdti.com/bdtimark/chip_float_scores.pdf

[12] Analog Devices, Inc. (2006). *Embedded Processing & DSP*. [Online]. Available: http://www.analog.com/IST/SelectionTableProcessors/?selection_table_id=%67&Cat=Processors&SubCat=SHARC%20Processor

[13] D. E. Corporation, *Digital Semiconductor Alpha 21064 and Alpha 21064A Microprocessors Hardware Reference Manual*, Jun. 1996, Maynard, MA: Digital Equipment Corporation.

[14] D. Abbott, *PCI Bus Demystified*, 2nd ed. Burlington, MA: Elsevier, 2004.

Charles B. Cameron (S'88–M'89–SM'05) received the B.Sc. degree in computer science from the University of Toronto, Toronto, ON, Canada, in 1997 and the M.S.E.E. and Ph.D. degrees in electrical engineering from the Naval Postgraduate School, Monterey, CA, in 1989 and 1991, respectively.

He is a commander in the U.S. Navy. He qualified as a Mission Commander in E-2C Hawkeye Airborne early warning aircraft in 1983. He is currently an Assistant Professor at the United States Naval Academy, Annapolis, MD, and a Lecturer at the Johns Hopkins University, Laurel, MD. His research and teaching interests are in computer architecture, reconfigurable computing, and sensors. He holds a patent for an electronic demodulator used to recover signals from interferometric fiber-optic sensors.

Rosa Nívea Rodríguez (M'04) was born in Puerto Rico in 1982. She received the B.S. degree in computer engineering from the Universidad de Puerto Rico, Mayaguez, Puerto Rico, in May 2004.

She served as an Intern at the NASA Goddard Space Flight Center, Greenbelt, MD, during the summer of 2002.

Ms. Rodriguez was the President of the Computer Society student chapter at the Mayaguez Campus of the Universidad de Puerto Rico and is a Member of the Society of Women Engineers (SWE) and the Tau Beta Pi Honor Society.

Nathan Padgett was born in Washington DC, in 1982. He is currently a senior student majoring in computer science at the Georgia Institute of Technology, Atlanta.

He is specializing in graphics, artificial intelligence, and the human–computer interface. Once his career as a pilot is over, he hopes to use those specialties to help create the cockpits of the future and to work on the unmanned flight programs currently being implemented by the Air Force.

Mr. Padgett is a member of Delta Tau Delta Fraternity at Georgia Institute of Technology. He is also in the Air Force Reserve Officer Training Corps (ROTC) and is slated to attend undergraduate pilot training upon graduation in May 2005.

Eugene Waluschka received the Ph.D. degree in physics from New York University, New York, NY, in 1975.

He joined the Mathematical Applications Group Elmsford, NY, in 1976 to work in the areas of nuclear radiation transport by Monte Carlo methods, simulation of aircraft laser signatures, and computer graphics. In 1980, he joined the PerkinElmer Corporation, Shelton, CT, where most of his time was spent in modeling and numerically simulating various phenomena. Currently, he is with the National Aeronautics and Space Administration's (NASA's) optics group at Goddard Space Flight Center (GSFC), Greenbelt, MD, in the optics group, supporting the Laser Interferometer Space Antenna (LISA) gravitational-wave detection mission, the Constellation-X ray telescope, and the earth observing missions.

Semion Kizhner received the M.S. degree in computer science from Johns Hopkins University, Baltimore, MD, in 1980.

He is now an Aerospace Engineer with the National Aeronautics and Space Administration (NASA) at the Goddard Space Flight Center (GSFC), Greenbelt, MD. He participated in the development of the space shuttle-launched hitchhiker carrier and several attached shuttle payloads such as the robot operated materials processing system (ROMPS). He was responsible for establishing the global-positioning-system (GPS) applications and test facility at GSFC and supported GPS simulations for several space projects, such as the OrbView-2, SAC-A, and EO-1 spacecraft. He is currently developing capabilities to access spacecraft as nodes on the Internet, to accelerate generation of images derived from weather spacecraft data, and implementing algorithms for high-rate control loops in optical instruments using images. He proposed the development of the Hilbert–Huang transform data processing system for spectrum analysis of nonlinear and nonstationary data and has been leading the development team.

Gabriel Colón is a student in electrical engineering at the Mayagüez Campus, Universidad de Puerto Rico (UPRM), where he has been on the Dean's List.

He was accepted to work at National Aeronautics and Space Administration's (NASA's) Goddard Space Flight Center (GSFC), Greenbelt, MD, during the summer of 2003 and at the Applied Physics Laboratory of the Johns Hopkins University, Baltimore, MD, during the summer of 2004.

Mr. Colón is also a former member of the Golden Key, and served as Vice-President in 2003–2004. He was selected for the Partnership for Spatial and Computational Research Program (sponsored by NASA and UPRM), the United States Achievement Academy Award, and the All-American Scholar Collegiate Award while still a student.

Colleen Weeks received the secondary-level education from Catoctin High School, Thurmont, MD, in the spring of 2004 and is currently a freshman at the Johns Hopkins University, Baltimore, MD.

She is a former National Space Club Scholar who interned at National Aeronautics and Space Administrations (NASA) Goddard Space Flight Center (GSFC), Greenbelt, MD, during the summer of 2003, where she assisted the Microelectronics and Signal Processing Branch.