©2004 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Optical Ray Tracing Using Parallel Processors

Charles B. Cameron, *Member, IEEE*, Rosa Nívea Rodríguez, *Member, IEEE*, Nathan Padgett, Eugene Waluschka, and Semion Kizhner

***Abstract*—One of the instruments on the sun-synchronous Terra (EOS AM) and Aqua (EOS PM) spacecraft, the Moderate Resolution Spectroradiometer (MODIS), obtains calibration data once during every orbit. Observations of the sun permit corrections to observations of the earth during the ensuing orbit. Although the instrument was designed to receive uniform sunlight over the entire surface of its detector, the sunlight was in fact not uniform. While this did not adversely affect the calibration, it nonetheless implied a lack of understanding of how the optical system really functioned. To learn what was wrong, NASA used an optical ray-tracing program on a DEC Alpha computer. The results correlated well with the observations made by the instrument itself, but it took nearly two weeks to complete the computer simulation, a discouragingly long time. This paper describes the algorithm and its implementation in a system with multiple digital signal processor (DSP) chips operating in parallel. Timing data show a highly linear relationship between the number of DSPs present and the speed of the computation. Administrative overhead is negligible compared to the time taken to compute ray trajectories. This implies that many more than just four DSPs could be harnessed before administrative overhead would begin to be significant.**

***Index Terms*—Digital signal processor (DSP), optical ray tracing, optics, parallel processing, reconfigurable computer (RC), reconfigurable computing.**

I. Introduction

IN ORDER to speed up optical ray tracing simulations, the authors investigated the implementation of ray-tracing algorithms in a PC-based system with multiple digital signal processor (DSP) microprocessors within. Doing these computations took about two weeks on a particular uniprocessor system. This paper:

1) explains the mathematics associated with optical ray tracing;
2) explores the use of multiple coordinate systems to simplify such mathematics;
3) discusses how we divided the work up into tasks which can be executed in parallel on multiple processors; and
4) presents measurements showing how speed of processing increases nearly linearly with an increase in the number of processors available.

Rather than simply use a supercomputer, we investigated the use of low-cost, reconfigurable computer (RC) architectures and digital signal processors.

Manuscript received May 30, 2003; revised June 23, 2004. This work was supported by NASA and by the United States Naval Academy.

C. B. Cameron is with the United States Naval Academy, Annapolis, MD 21402 USA.

R. N. Rodríguez is with Guidant Corporation, St. Paul, MN 55112-5798 USA.

N. Padgett is a student at the Georgia Institute of Technology, Atlanta, GA 30332 USA.

E. Waluschka and S. Kizhner are with NASA Goddard Space Flight Center, Greenbelt, MD 20768 USA.

Digital Object Identifier 10.1109/TIM.2004.838131

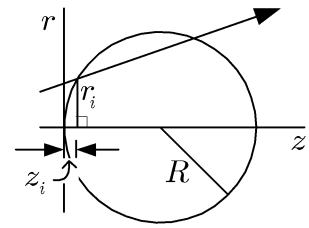

Fig. 1. Spherical Lens.

Notes on the mathematical notation used appear in the Appendix.

II. Optics

Ray tracing is one of the principle tools for an optical system designer [1]. Computer-based ray tracing has a history going back for more than 40 years. An early paper presenting a unified ray-tracing procedure that is applicable to optical systems with very general kinds of surfaces is Spencer and Murty [2]. Their procedures rely on matrix methods offering a compact notation. While some other authors also use matrix-based methods [3], others avoid them [4], [5]. Of those using matrix-based methods, some use two-dimensional (2–D) matrix methods [6] while others use three-dimensional (3–D) matrix methods [7]. In this section, we summarize the 3-D matrix-based equations for optical reflection and refraction of rays impinging on various simple geometric surfaces as developed in [7]. The notation is somewhat simplified here for greater clarity.

A. Optical Surfaces

While in principle, lenses can be made with any arbitrary surface, for simplicity, we usually consider only simple geometric shapes: planes and conicoids (spheres, paraboloids, ellipsoids, and hyperboloids).[1]

1) A Spherical Lens: Fig. 1 shows a crosssection of a sphere. Not shown are the x- and y-axes, orthogonal to the z-axis and to each other. This sphere may be regarded as a lens whose vertex is at point $(0, 0, 0)$ and which is centered around the positive z-axis. For such a spherical lens of radius R

$$R^2 = x^2 + y^2 + (z - R)^2. \tag{1}$$

If we define

$$r^2 = x^2 + y^2 \tag{2}$$

[1]Paraboloids, ellipsoids, and hyperboloids are the surfaces described by parabolas, ellipses, and hyperbolas rotated about an axis of symmetry. In the case of hyperbolas, the axis is usually considered to be the longitudinal or semimajor axis. In the case of ellipses, either axis may be considered.

0018-9456/$20.00 © 2005 IEEE

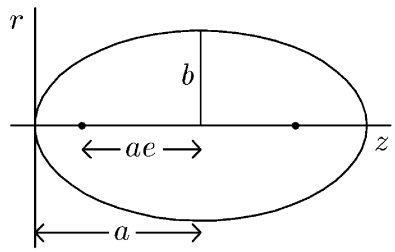

Fig. 2. Prolate Ellipsoid. The semimajor axis is a, the semiminor axis is b, and the eccentricity is e.

and let the curvature $c = 1/R$, then it can be shown that for a sufficiently small value of r

$$z \approx \frac{1}{2}cr^2. \tag{3}$$

In other words, the surface of a sphere can be approximated by a parabolic surface. This approximation is only reasonable if $r \ll R$, which is the case for a typical spherical lens.

2) Prolate Ellipsoid Lens and Other Conicoid Lenses: Fig. 2 shows a cross section of a prolate ellipsoid[2] with eccentricity e, semi-major axis a, and semi-minor axis b. Variable r is given by $r^2 = x^2 + y^2$ as in (2).

An equation of this ellipsoid is

$$\frac{(z-a)^2}{a^2} + \frac{r^2}{b^2} = 1. \tag{4}$$

We can define the eccentricity e of the ellipse implicitly by

$$b^2 = a^2(1-e^2) \tag{5}$$

and if we define

$$\varepsilon = 1 - e^2 \tag{6}$$

and

$$c = \frac{1}{a\varepsilon} \tag{7}$$

then it can be shown that the least positive position z (leftmost in Fig. 2) at which the surface of the ellipsoid is a radius r from the z-axis is given by

$$z = \frac{cr^2}{1 + \sqrt{1 - \varepsilon(cr)^2}}. \tag{8}$$

If we define the conic constant

$$k = \varepsilon - 1 = \frac{b^2}{a^2} - 1 = -e^2 \tag{9}$$

then we can rewrite (8) as

$$z = \frac{cr^2}{1 + \sqrt{1 - (1+k)c^2r^2}}. \tag{10}$$

It can be shown (see [7]) that this equation is applicable not just for prolate ellipsoids but for all conicoids. It can be expanded in a power series using a Taylor series about $r = 0$, giving the same result given earlier in (3) for a spherical surface. In other words, a parabolic surface can approximate both a spherical surface and a prolate ellipsoidal surface. The table provided in [7] and reproduced here as Table I shows how to pick the conic constant k for various conicoids.

[2]Prolate with respect to the optical axis z.

TABLE I
PARAMETERS OF THE CONICOIDS

Eccentricity	Conic Constant	ε	Conic Section
	$k > 0$	$\varepsilon > 1$	oblate ellipsoid
$e = 0$	$k = 0$	$\varepsilon = 1$	sphere
$0 < e < 1$	$-1 < k < 0$	$0 < \varepsilon < 1$	prolate ellipsoid
$e = 1$	$k = -1$	$\varepsilon = 0$	paraboloid
$e > 1$	$k < -1$	$\varepsilon < 0$	hyperboloid

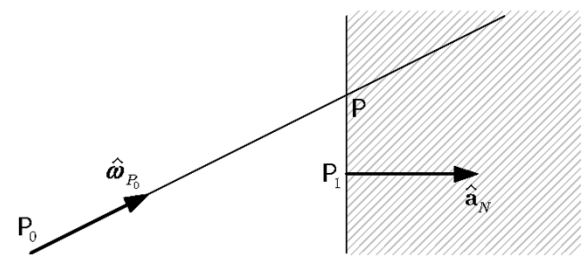

Fig. 3. Intersection of a line with a plane.

B. Ray Intersections

To trace a ray's path requires the following two steps.

1) Find the point where a ray intersects an optical surface.
2) Find the direction the ray takes after striking the surface.

In this section, we consider the first of these problems for various shapes of optical surfaces.

1) Intersection of a Line With a Plane: Consider Fig. 3. It shows a ray departing initial point P_0 with direction vector $\hat{\boldsymbol{\omega}}_{P_0}$. Unless it is parallel to (or pointing away from) the plane depicted, it will strike the plane at point P, whose coordinates we wish to determine. The plane can be defined by its unit-length normal $\hat{\boldsymbol{a}}_N$ and a particular point P_1 in the plane. It can be shown that

$$\boldsymbol{P} = \boldsymbol{P}_0 + \frac{\hat{\boldsymbol{a}}_N \cdot (\boldsymbol{P}_1 - \boldsymbol{P}_0)}{\hat{\boldsymbol{a}}_N \cdot \hat{\boldsymbol{\omega}}_{P_0}} \hat{\boldsymbol{\omega}}_{P_0}. \tag{11}$$

If P_0 is already in the plane, then $\hat{\boldsymbol{a}}_N \cdot (\boldsymbol{P}_1 - \boldsymbol{P}_0) = 0$ and obviously in this case $\boldsymbol{P} = \boldsymbol{P}_0$. If the ray is parallel to the plane, then $\hat{\boldsymbol{a}}_N \cdot \hat{\boldsymbol{\omega}}_{P_0} = 0$, resulting in division by zero if (12) is applied. A computer program must check for this condition before applying the equation.

In the special case where we choose our coordinate system so that $\mathsf{P}_1 = (0, 0, 0)$, we can write (11) as

$$\boldsymbol{P} = \boldsymbol{P}_0 - \frac{\hat{\boldsymbol{a}}_N \cdot \boldsymbol{P}_0}{\hat{\boldsymbol{a}}_N \cdot \hat{\boldsymbol{\omega}}_{P_0}} \hat{\boldsymbol{\omega}}_{P_0}. \tag{12}$$

It should be noted that even if the ray is pointing away from the plane, (11) and (12) will find the intersection of the line (collinear with the ray) and the plane. A computer program must check to see if this is the case.

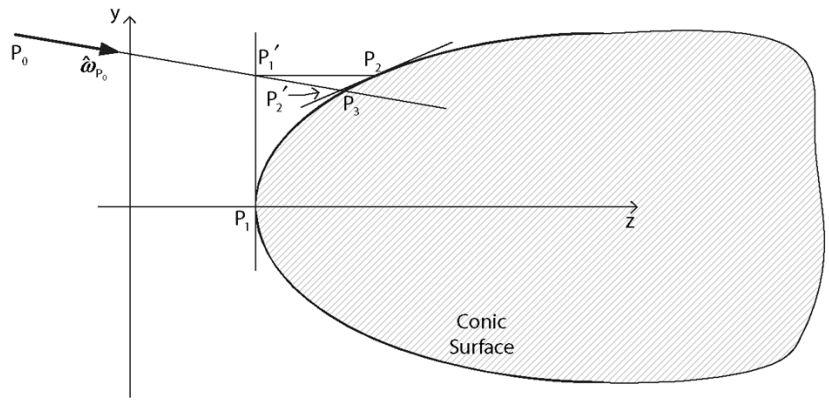

Fig. 4. Intersection of a line with a conic surface.

2) Intersection of a Line With a Conic Surface: From (2) and (10), a general conic surface, a conicoid, can be described by an equation of the form

$$g(x,y,z) = z - \frac{c(x^2+y^2)}{1+\sqrt{1-(1+k)c^2(x^2+y^2)}}. \tag{13}$$

Such a surface and a line originating at point P_0 and intersecting the surface are shown in Fig. 4. The direction of the line is given by the unit vector $\hat{\boldsymbol{\omega}}_{P_0} = L\hat{\boldsymbol{u}}_x + M\hat{\boldsymbol{u}}_y + N\hat{\boldsymbol{u}}_z$.

We can follow an iterative procedure as indicated in the figure to discover the point where the line intersects the surface. We do this by finding a series of points of intersection between the line and surfaces tangent to the conic surface.

Although Fig. 4 implies that the y-axis does not intersect the conic, we are free to define any coordinate system we like. It is often convenient to choose the xy-plane as being tangent to the vertex of the conic surface. If we do this, then the point $\mathsf{P}_1 = (0,0,0)$ becomes a known point in the first plane considered. This plane has normal $\hat{\boldsymbol{a}}_{N1} = \hat{\boldsymbol{u}}_z$. Using (11), we can find the point P_1' where the line intersects this plane.

$$\boldsymbol{P}_1' = \boldsymbol{P}_0 + \frac{\hat{\boldsymbol{a}}_{N1}\cdot(\boldsymbol{P}_1-\boldsymbol{P}_0)}{\hat{\boldsymbol{a}}_{N1}\cdot\hat{\boldsymbol{\omega}}_{P_0}}\hat{\boldsymbol{\omega}}_{P_0}. \tag{14}$$

In particular, we can find the values of x and y of the point of intersection. (In the figure, $x = 0$, but in general, it could be anything.)

Next, we find a point P_2 with the same values of x and y but lying on the conic surface. This point satisfies the equation $g(x,y,z) = 0$, so it has the coordinates

$$\mathsf{P}_2 = \left(x_1', y_1', \frac{c({x_1'}^2+{y_1'}^2)}{1+\sqrt{1-(1+k)c^2\left({x_1'}^2+{y_1'}^2\right)}}\right). \tag{15}$$

After this, we find the plane tangent to the conic surface at point P_2. This requires finding the partial derivatives of g and evaluating them at point P_2. If we define

$$a = \sqrt{1-(1+k)c^2(x^2+y^2)} \tag{16}$$

$$b = \frac{c}{a(1+a)^2}\left(2a(1+a)+c^2(1+k)\left(x^2+y^2\right)\right) \tag{17}$$

and

$$d = \sqrt{(1/b)^2+x_2^2+y_2^2} \tag{18}$$

then it can be shown that the direction cosines of the normal to the tangent plane are

$$\alpha_2 = -\frac{x_2}{d} \tag{19}$$

$$\beta_2 = -\frac{y_2}{d} \tag{20}$$

$$\gamma_2 = +\frac{1}{bd} \tag{21}$$

and the unit-length normal to the tangent plane at point P_2 is given by

$$\hat{\boldsymbol{a}}_{N2} = \alpha_2\hat{\boldsymbol{u}}_x + \beta_2\hat{\boldsymbol{u}}_y + \gamma_2\hat{\boldsymbol{u}}_z. \tag{22}$$

This procedure can be repeated: from the intersection of the extended ray with a plane tangent to the conic, we can determine a nearby point on the conic, and from this we can determine a new tangent plane. The entire procedure is repeated until successive points are close enough together to be regarded as equivalent.

Fig. 5 summarizes the steps in finding the intersection between a line and a conic surface.

3) Intersection of a Ray and a Spherical Surface: A sphere is a special case of a conic, of course, but there is a simpler method of obtaining the point of intersection of a ray and a spherical surface, one that avoids the iteration shown above. Fig. 6 shows a ray departing initial point P_0 with direction vector $\hat{\boldsymbol{\omega}}_{\boldsymbol{P}_0}$ and striking point P. It can be shown that the steps in Fig. 7 let us find the coordinates of point P and the sphere's centrally directed normal $\hat{\boldsymbol{a}}_N$ at this point.

C. Reflections

Now that we know how to discover the point on a plane or a conic surface where a ray strikes it, we can turn our attention to what happens to the ray next. In a typical optical system, part of the ray is reflected and part is refracted, but this is a complication which we shall neglect. For a reflective surface, we shall disregard the refracted portion, and for a refractive surface, we shall disregard the reflected portion.[3]

1) Reflection: Fig. 8 shows a unit-length ray $\hat{\boldsymbol{\omega}}_{\boldsymbol{P}_0}$ originating at initial point P_0 and incident on a surface S at point O. Its reflection, unit-length ray $\hat{\boldsymbol{\omega}}_{P_1}$, terminates at point P_1. The

[3]Including these contributions would greatly increase the number of rays traced but would increase the accuracy of the model.

For a ray with direction $\hat{\boldsymbol{\omega}}_{P_0} = L\hat{\boldsymbol{u}}_x + M\hat{\boldsymbol{u}}_y + N\hat{\boldsymbol{u}}_z$ originating at point $\mathsf{P}_0 = (x_0, y_0, z_0)$ set $\boldsymbol{P}_1 = 0$, $\hat{\boldsymbol{a}}_{N_1} = \hat{\boldsymbol{a}}_z$, $j = 1$.

1. Calculate
$$\boldsymbol{P}'_j = \boldsymbol{P}_0 + \frac{\hat{\boldsymbol{a}}_{Nj} \cdot (\boldsymbol{P}_j - \boldsymbol{P}_0)}{\hat{\boldsymbol{a}}_{Nj} \cdot \hat{\boldsymbol{\omega}}_{P_0}} \hat{\boldsymbol{\omega}}_{P_0}. \qquad \text{(from (11))}$$
2. Set $j = j + 1$.
3. Set point
$$\mathsf{P}_j = \left(x'_{j-1}, y'_{j-1}, \frac{c\left({x'_{j-1}}^2 + {y'_{j-1}}^2\right)}{1 + \sqrt{1 - (1+k)c^2\left({x'_{j-1}}^2 + {y'_{j-1}}^2\right)}} \right). \qquad \text{(from (15))}$$
4. Calculate
$$a = \sqrt{1 - (1+k)c^2\left(x_j^2 + y_j^2\right)} \qquad \text{(from (16))}$$
$$b = \frac{c}{a\,(1+a)^2}\left(2a(1+a) + c^2(1+k)\left(x_j^2 + y_j^2\right)\right) \qquad \text{(from (17))}$$
$$d = \sqrt{(1/b)^2 + x_j^2 + y_j^2}. \qquad \text{(from (18))}$$
5. Calculate
$$\alpha_j = -\frac{x_j}{d}, \quad \beta_j = -\frac{y_j}{d}, \quad \gamma_j = +\frac{1}{bd}. \qquad \text{(from (19), (20), (21))}$$
6. Let $\hat{\boldsymbol{a}}_{Nj} = \alpha_j \hat{\boldsymbol{u}}_x + \beta_j \hat{\boldsymbol{u}}_y + \gamma_j \hat{\boldsymbol{u}}_z$.
7. Repeat until P_j and P'_j differ by an insignificant amount.

Fig. 5. Summary: Finding the intersection of a ray and a conic surface.

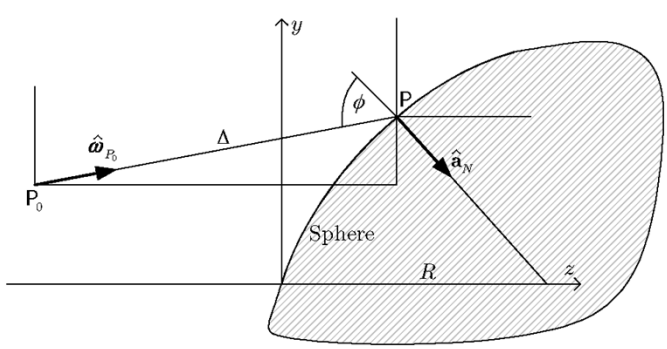

Fig. 6. Ray tracing for a spherical lens.

$$\hat{\boldsymbol{\omega}}_{P_0} = L\hat{\boldsymbol{a}}_x + M\hat{\boldsymbol{a}}_y + N\hat{\boldsymbol{a}}_z \tag{23}$$
$$c = 1/R \tag{24}$$
$$F = c\left(x_A^2 + y_A^2 + z_A^2\right) - 2z_A \tag{25}$$
$$G = N - c\left(x_A L + y_A M + z_A N\right) \tag{26}$$
$$\Delta = \frac{F}{G \pm \sqrt{G^2 - cF}} \tag{27}$$

(Select the root which yields Δ with minimum magnitude.)

$$\boldsymbol{P} = \boldsymbol{P}_0 + \Delta \cdot \hat{\boldsymbol{\omega}}_{P_0} \tag{28}$$
$$\hat{\boldsymbol{a}}_N = -c\,x_B\,\hat{\boldsymbol{a}}_x - c\,y_B\,\hat{\boldsymbol{a}}_y + (1 - c\,z_B)\hat{\boldsymbol{a}}_z \tag{29}$$

Fig. 7. Summary: Steps to find the where a ray strikes a spherical surface and the normal at that point.

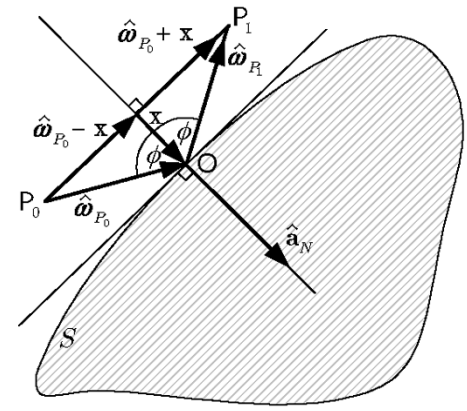

Fig. 8. Reflection of a ray at a surface.

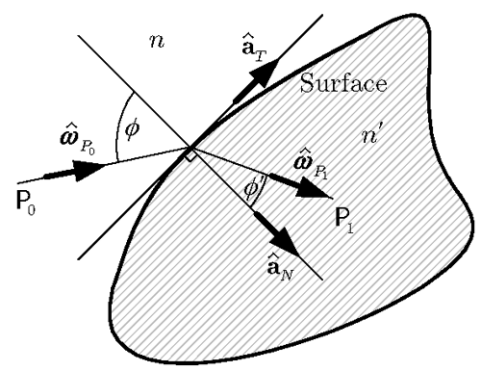

Fig. 9. Vector form of Snell's Law.

angle of incidence and the angle of reflection both are ϕ. Given the centrally directed unit-length normal $\hat{\boldsymbol{a}}_N$ to the surface, it can be shown that

$$\hat{\boldsymbol{\omega}}_{P_1} = \hat{\boldsymbol{\omega}}_{P_0} - 2(\hat{\boldsymbol{\omega}}_{P_0} \cdot \hat{\boldsymbol{a}}_N)\hat{\boldsymbol{a}}_N. \tag{30}$$

2) Refraction: Fig. 9 shows a light ray $\hat{\boldsymbol{\omega}}_{P_0}$ traveling through a medium with index of refraction n. The ray passes through a surface into a different medium with index of refraction n' with angle of incidence ϕ. The ray emerges as $\hat{\boldsymbol{\omega}}_{P_1}$ from the surface at a different angle, ϕ', which we desire to calculate from n, n', and ϕ.

Unit vector $\hat{\boldsymbol{a}}_N$ gives the direction of the normal to the surface, coplanar with $\hat{\boldsymbol{\omega}}_{P_0}$ and $\hat{\boldsymbol{\omega}}_{P_1}$, and pointing into the new medium; and $\hat{\boldsymbol{a}}_T$ is a unit vector tangent to the surface such that

$$\hat{\boldsymbol{\omega}}_{P_0} = \cos\phi\boldsymbol{a}_N + \sin\phi\boldsymbol{a}_T \tag{31}$$
$$\hat{\boldsymbol{\omega}}_{P_1} = \cos\phi'\boldsymbol{a}_N + \sin\phi'\boldsymbol{a}_T. \tag{32}$$

With these definitions, it can be shown that

$$\hat{\boldsymbol{\omega}}_{P_1} = \frac{n}{n'}\hat{\boldsymbol{\omega}}_{P_0} + \hat{\boldsymbol{a}}_N\left(\cos\phi' - \frac{n}{n'}\cos\phi\right). \tag{33}$$

3) Efficient Computation: We would like to be able to compute $\hat{\boldsymbol{\omega}}_{P_1}$ from (33) efficiently if we are going to compute it repeatedly. Calculating any trigonometric function is time-consuming so we would prefer to avoid it, if possible.

It can be shown that if we let $q = n/n'$, then

$$\cos\phi = \hat{\boldsymbol{a}}_N \cdot \hat{\boldsymbol{\omega}}_{P_0} \tag{34}$$

$$\cos\phi' = \pm\sqrt{1 - q^2(1 - \cos^2\phi)}. \tag{35}$$

Therefore, it suffices to compute

$$\hat{\boldsymbol{\omega}}_{P_1} = q\hat{\boldsymbol{\omega}}_{P_0} + \hat{\boldsymbol{a}}_N(\cos\phi' - q\cos\phi) \tag{36}$$

with $\cos\phi$ given by (34) and $\cos\phi'$ given by (35). Although we still must compute a square root, this ordinarily requires considerably less time than computing trigonometric functions. This is especially true with our hardware setup because the AD21160 contains special instructions to reduce the time to compute square roots.

Should we choose the positive or the negative square root? We should pick it so the arithmetic sign of $\cos\phi'$ is the same as that of $\cos\phi$. Commonly, $\phi < \pi/2$ and $\phi' < \pi/2$, so both trigonometric functions are positive.

III. Converting Between Different Coordinate Systems

Ray tracing with respect to a particular surface is simplified if all computations are performed in a local coordinate system. In coordinate system k associated with surface k, point P_A is represented by vector $\boldsymbol{r}_{Ak}$, point P_B is represented by vector $\boldsymbol{r}_{Bk}$, and the ray departing point A is $\hat{\omega}_{Ak}$. In a system with n surfaces, the procedure for each surface $k \in [0, n-1]$ is as follows.

1) Find the point P_B on surface k struck by the ray $\hat{\boldsymbol{\omega}}_{Ak}$ originating at point P_A. If the ray does not strike the surface, it can be discarded. This is tantamount to disregarding other forms of internal ray propagation, such as scattering.
2) Find the ray $\hat{\boldsymbol{\omega}}_{Bk}$ resulting from the interaction of the ray and the surface, whether due to reflection or to refraction.
3) Convert vectors $\boldsymbol{r}_{Bk}$ and $\hat{\boldsymbol{\omega}}_{Bk}$ from the coordinate system of optical medium k (applicable before interaction with surface k) to the equivalent vectors $\boldsymbol{r}_{A,k+1}$ and $\hat{\boldsymbol{\omega}}_{A,k+1}$ of optical medium $k+1$ (applicable after interaction with surface k).

To carry out these steps requires knowing how to do the coordinate conversions.

Suppose the origin of local frame k is located in frame 0 (the global frame) at point T_k, represented by a vector $\boldsymbol{t}_k = \delta_{xk}\hat{\boldsymbol{u}}_x + \delta_{yk}\hat{\boldsymbol{u}}_y + \delta_{zk}\hat{\boldsymbol{u}}_z$. To convert vectors expressed in local frame k to equivalent vectors in the global reference frame, frame 0, requires first rotating them through angle γ_k around the $+z$-axis, through angle $-\beta_k$ around the y-axis, and last, through angle $-\alpha_k$ around the x-axis; and then translating them to reflect the fact that the two frames do not share the same origin.[4]

A general point P can be represented in local frame k by a vector $\boldsymbol{r}_k = x_k\hat{\boldsymbol{u}}_x + y_k\hat{\boldsymbol{u}}_y + z_k\hat{\boldsymbol{u}}_z$. The same point can be represented in frame 0 (the global frame) by a vector $\boldsymbol{r}_0 = \mathbf{A}_k^{-1}\boldsymbol{r}_k + \boldsymbol{t}_k$ where matrix $\mathbf{A}_k^{-1}$ is given by

$$\mathbf{A}_k^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_k & \sin\alpha_k \\ 0 & -\sin\alpha_k & \cos\alpha_k \end{pmatrix} \cdot \begin{pmatrix} \cos\beta_k & 0 & -\sin\beta_k \\ 0 & 1 & 0 \\ \sin\beta_k & 0 & \cos\beta_k \end{pmatrix} \cdot \begin{pmatrix} \cos\gamma_k & -\sin\gamma_k & 0 \\ \sin\gamma_k & \cos\gamma_k & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Now consider another frame of reference, frame $k+1$, located in frame 0 at point T_{k+1}, represented by a vector $\boldsymbol{t}_{k+1} = \delta_{x,k+1}\hat{\boldsymbol{u}}_x + \delta_{y,k+1}\hat{\boldsymbol{u}}_y + \delta_{z,k+1}\hat{\boldsymbol{u}}_z$. To convert vectors expressed in the global reference frame, frame 0, to equivalent vectors in local frame $k+1$ requires first translating them to the origin and then rotating them through angle α_{k+1} around the x-axis, through angle β_{k+1} around the y-axis, and finally through angle $-\gamma_{k+1}$ around the $+z$-axis

$$\begin{aligned} \boldsymbol{r}_{k+1} &= \mathbf{A}_{k+1}(\boldsymbol{r}_0 - \boldsymbol{t}_{k+1}) \\ &= \mathbf{A}_{k+1}(\mathbf{A}_k^{-1}\boldsymbol{r}_k + \boldsymbol{t}_k - \boldsymbol{t}_{k+1}) \\ &= (\mathbf{A}_{k+1}\mathbf{A}_k^{-1})\boldsymbol{r}_k + \mathbf{A}_{k+1}(\boldsymbol{t}_k - \boldsymbol{t}_{k+1}) \\ \boldsymbol{r}_{k+1} &= \mathbf{A}_{k+1,k}\boldsymbol{r}_k + \boldsymbol{t}_{k+1,k} \end{aligned} \tag{37}$$

where

$$\mathbf{A}_{k+1} = \begin{pmatrix} \cos\gamma_{k+1} & \sin\gamma_{k+1} & 0 \\ -\sin\gamma_{k+1} & \cos\gamma_{k+1} & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\beta_{k+1} & 0 & \sin\beta_{k+1} \\ 0 & 1 & 0 \\ -\sin\beta_{k+1} & 0 & \cos\beta_{k+1} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_{k+1} & -\sin\alpha_{k+1} \\ 0 & \sin\alpha_{k+1} & \cos\alpha_{k+1} \end{pmatrix} \tag{38}$$

$$\mathbf{A}_{k+1,k} = \mathbf{A}_{k+1}\mathbf{A}_k^{-1} \tag{39}$$

and

$$\boldsymbol{t}_{k+1,k} = \mathbf{A}_{k+1}(\boldsymbol{t}_k - \boldsymbol{t}_{k+1}). \tag{40}$$

Using (37), we can calculate the vector representation $\boldsymbol{r}_{k+1}$ of point P in frame $k+1$ if we know its vector representation $\boldsymbol{r}_k$ in frame k. Since all rotation angles are fixed in a given lens system, the matrices $\mathbf{A}_{k+1,k}$ and the vectors $\boldsymbol{t}_{k+1}$ can be computed in advance—they need not be recomputed on the fly each time. This means that ray tracing amounts to a sequence of matrix multiplications, vector additions, and calculations of intersection points as well as reflections or refractions.

IV. Ray Tracing in the MODIS Instrument

This section explains why standard ray-tracing algorithms and software products are not appropriate for NASA's purposes.

A. MODIS Optical Subsystem

Fig. 10 shows how light reaches the optical detector in the MODIS instrument. Rays of light from the disk of the sun first strike an attenuator. The purpose of the attenuator is to reduce the optical power of the sun's rays from 25 W to about 8.5% of

[4]A positive angle is that required, say, to rotate $\hat{\boldsymbol{u}}_x$ to align with $\hat{\boldsymbol{u}}_y$ in a right-handed system. The conventional sequence of rotations described above is not followed universally. It is used in the treatment by Kidger [7] summarized here and is also used in the commercial ray-tracing program CODE V from Optical Research Associates. It is essential to be clear about the convention in use before the mathematics will make any sense at all.

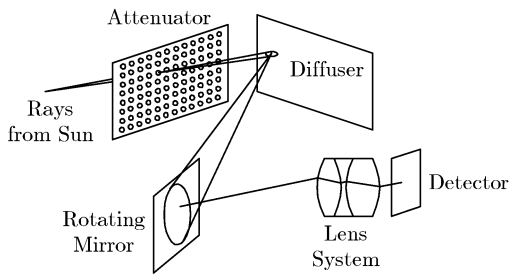

Fig. 10. MODIS optical system on the Terra and Aqua spacecraft.

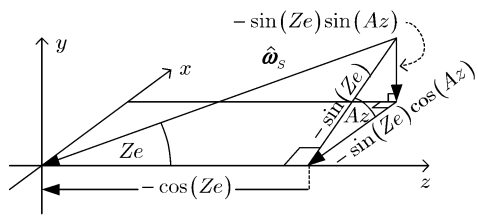

Fig. 11. Direction cosines of a unit-length ray from the sun to MODIS.

this, or around 2 W, in order to perform instrument calibration without damaging the detector.

Fig. 11 shows a unit-length ray $\hat{\boldsymbol{\omega}}_S$ from the sun arriving at the MODIS instrument. The ray makes a zenith angle Ze with the z-axis of the MODIS instrument. The projection of the ray onto the xy-plane makes an azimuth angle Az with the x-axis. In Fig. 11, the ray's projections on the x-, y-, and z-axes all are shown. The ray can be decomposed in the frame of reference of the MODIS instrument

$$\begin{aligned}\hat{\boldsymbol{\omega}}_S &= \omega_x\hat{\boldsymbol{u}}_x + \omega_y\hat{\boldsymbol{u}}_y + \omega_z\hat{\boldsymbol{u}}_z \\ &= -\sin(Ze)\cos(Az)\hat{\boldsymbol{u}}_x - \sin(Ze)\sin(Az)\hat{\boldsymbol{u}}_y \\ &\quad + \cos(Ze)\hat{\boldsymbol{u}}_z \end{aligned} \tag{41}$$

where $\hat{\boldsymbol{u}}_x, \hat{\boldsymbol{u}}_y$, and $\hat{\boldsymbol{u}}_z$ are unit vectors in the x-, y-, and z-directions, respectively. The quantities ω_x, ω_y, and ω_z are the direction cosines of the ray. At the time a calibration is performed, we take the zenith angle $Ze = 77.521°$ and the azimuth angle $Az = -24.560°$. These are the correct values as Terra passes over the north pole once each orbit in its sun-synchronous orbit.

The ray then passes through the numerous pinholes of the attenuator, a perforated metal screen, as described by Waluschka *et al.* [8]. These pinholes each have a radius of 1 mm and their edges are tapered to reduce the effects of the thickness of the screen. The pinholes are arranged in a grid of scalene triangles in order to provide even illumination through the surfaces which follow the rotating mirror.

To be more precise, a cone of rays passes through each pinhole, due to the fact that the sun is not a point source but has a disk of apparent half-angle $\theta_S = 0.25°$[9].[5] Despite the fact that each pinhole has a sizable diameter, we shall treat it as a point source of a conical fan of rays. We shall also neglect the fact that the intensity of the incident light is diminished by its being spread, since all pinholes result in a comparable diminution of intensity.

As described in [8], each cone of light from a pinhole in the screen strikes the diffusing surface. Each ray within each cone, then, gives rise to another cone of rays from the diffusing surface. We shall treat this diffusing surface as a Lambertian surface, meaning that its radiance is independent of direction. Each diffusely reflected ray can, therefore, be regarded as having the same radiance. Consequently, we need not keep track of the radiance associated with each ray.

After departing the diffusing surface, the rays enter an optical system. It consists of an initial reflecting plane mirror which can be rotated out of the way when images of earth are taken. Following it are a series of reflecting and refracting surfaces. All told, there are 29 surfaces preceding and following the mirror. They are numbered S_k where $k \in [0, N-1]$. Surface 1 specifies a 177.8 mm eye pupil diameter. The attenuating screen is surface 2, the diffuser is surface 3, and the mirror is surface 4. Associated with surface S_k is the index of refraction n_k of the medium leading up to the surface. For simplicity, we regard the surfaces as producing no scattering, absorption, or reflection.

The task of tracing the rays through this series can be subdivided into the following repeated tasks, stated with reference to a unit-length ray $\hat{\boldsymbol{\omega}}_A$ in medium k departing from point A and heading for the surface S_{k+1} which separates medium k from medium $k+1$. Assume point A has coordinates $(x_{A_k}, y_{A_k}, z_{A_k})$ in the frame of reference of surface k. We can represent this point by the vector $\boldsymbol{A} = x_{A_k}\hat{\boldsymbol{u}}_{x_k} + y_{A_k}\hat{\boldsymbol{u}}_{y_k} + z_{A_k}\hat{\boldsymbol{u}}_{z_k}$. Assume also that the unit-length ray $\hat{\boldsymbol{\omega}}_A$ can be decomposed using known direction cosines into $\hat{\boldsymbol{\omega}}_A = \omega_{Ax_k}\hat{\boldsymbol{u}}_{x_k} + \omega_{Ay_k}\hat{\boldsymbol{u}}_{y_k} + \omega_{Az_k}\hat{\boldsymbol{u}}_{z_k}$. (The notation $\hat{\boldsymbol{u}}_{\alpha_k}$ refers to a unit vector in the direction of the α-axis in the coordinate system of surface S_k).

1) Find the position of point A in a new coordinate system, that of surface S_{k+1}. In the new coordinate system, $z = 0$ at the vertex of surface S_{k+1}. Point A has coordinates $(x_{A_{k+1}}, y_{A_{k+1}}, z_{A_{k+1}})$ in the new coordinate system and can be represented by vector $\boldsymbol{A} = x_{A_{k+1}}\hat{\boldsymbol{u}}_{x,k+1} + y_{A_{k+1}}\hat{\boldsymbol{a}}_{y,k+1} + z_{A_{k+1}}\hat{\boldsymbol{a}}_{z,k+1}$.
2) Find the direction cosines of unit-length ray $\hat{\boldsymbol{\omega}}_A$ in the new coordinate system. The ray can be expressed as $\hat{\boldsymbol{\omega}}_A = \omega_{Ax_{k+1}}\hat{\boldsymbol{u}}_{x_{k+1}} + \omega_{Ay_{k+1}}\hat{\boldsymbol{u}}_{y_{k+1}} + \omega_{Az_{k+1}}\hat{\boldsymbol{u}}_{z_{k+1}}$, where $\omega_{A\alpha_{k+1}}$ is the direction cosine with respect to the α-axis in the coordinate system of surface S_{k+1}.
3) Consider the plane P passing through the vertex of the surface S_{k+1}. Find the point A′ in plane P eventually reached by ray $\hat{\boldsymbol{\omega}}_{A_{k+1}}$. This point has coordinates $(x_{A'_{k+1}}, y_{A'_{k+1}}, 0)$ and can be represented by vector $\boldsymbol{A}' = x_{A'_{k+1}}\hat{\boldsymbol{u}}_{x_{k+1}} + y_{A'_{k+1}}\hat{\boldsymbol{u}}_{y_{k+1}}$.
4) Find the point B on surface S_{k+1} struck by ray $\hat{\boldsymbol{\omega}}_{A_{k+1}}$. This point has coordinates $(x_{B_{k+1}}, y_{B_{k+1}}, z_{B_{k+1}})$. The reason for finding the point A′ in plane P is to permit deciding whether or not the ray gets through an aperture associated with the surface. The aperture is regarded as lying in the xy-plane. Rays which fail to pass through the aperture successfully are discarded.
5) Find the new ray $\hat{\boldsymbol{\omega}}_B = \omega_{Bx_{k+1}}\hat{\boldsymbol{u}}_{x_{k+1}} + \omega_{By_{k+1}}\hat{\boldsymbol{u}}_{y_{k+1}} + \omega_{Bz_{k+1}}\hat{\boldsymbol{u}}_{z_{k+1}}$ by using (30) if this is a reflecting surface or Snell's law (32) if it is a refracting surface.

The point B with coordinates $(x_{B_{k+1}}, y_{B_{k+1}}, z_{B_{k+1}})$ and new ray $\hat{\boldsymbol{\omega}}_{B_{k+1}}$ in the coordinate system of surface S_{k+1} together represent the starting position and direction of a ray which now

[5]A more accurate value is $\theta_S = (1919\,\text{in}/2) = 0.2665°$

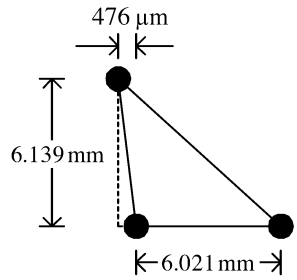

Fig. 12. Hole separation in the MODIS attenuator.

heads off toward surface S_{k+2}. The steps are repeated as many times as necessary to bring the ray to its final destination in the focal plane.

B. Hole Separation in the Attenuator

The pattern of hole separation on the MODIS instrument's attenuator is shown in Fig. 12. The attenuator is a rectangle whose half-width is w and whose half-height is h. The coordinates of the center of the attenuator are $(0, 0, 0)$ in its (local) coordinate system. There is a hole at the center and the horizontal spacing between holes is 6.0210 mm. The vertical spacing between holes is 6.1385 mm. Each row of holes is offset in the x-direction by -475.8 μm as y increases.

This pattern can be used to calculate the locations of all the holes on the attenuator by offsetting it vertically and horizontally throughout the extent of the attenuator. Triangles in the next row up will be offset by $\Delta = -475.8$ μm.

Regarding the 0th row as the one in which the base of the triangles is on $y = 0$, the horizontal offset Δ_j in any row j is $\Delta_j = j\Delta$, but since triangles shifted left or right by the horizontal spacing $w_r = 6.0210$ mm are indistinguishable, we can take the offset modulo this amount: $\Delta_j = j\Delta \bmod w_r$.

We can calculate the position of the most negative hole in row j by

$$x_{j,\min} = \left\lceil -\frac{w + \Delta_j}{w_r} \right\rceil w_r + \Delta_j. \tag{42}$$

The number n_j of holes which will fit in row j is given by

$$n_j = 1 + \left\lfloor \frac{2w - \Delta_j}{w_r} \right\rfloor. \tag{43}$$

So the kth hole in row j (where $k \in [0, n_j - 1]$) is at horizontal position

$$x_{jk} = x_{j,\min} + k\Delta_j \tag{44}$$

and that same hole is at vertical position

$$y_{jk} = kh_r. \tag{45}$$

V. Optical System Description File

In our implementation of ray tracing, we created an ASCII file to describe the optical system in the MODIS instrument. The format of this file was based loosely on the input format of the commercial ray-tracing program CODE V from Optical Research Associates. In this section, we list the elements in that file which describe the optical system.

A. General Items

Elements applicable to the entire system are as follows.

- Entrance pupil diameter: the diameter of the initial element of the optical system. Light rays falling outside this diameter do not enter the system.
- Wavelength: $\lambda_1[\lambda_2\, \lambda_3 \ldots]$ are the several wavelengths of light for which individual indices of refraction may be provided for successive lens elements.

B. Optical Surfaces in MODIS

Elements applicable to individual optical surfaces in the system are as follows.

- Curvature c of the lens element. If the curvature $c = 0$, then the surface is a plane; otherwise, it is treated as a conic surface.
- An indication showing whether the surface is a reflecting or a refracting surface.
- The indices of refraction $n_1[n_2\, n_3\, \ldots]$ for the surface, one for each wavelength specified for the system as a whole.
- Decentering specifications $x_{\mathrm{DS}}, y_{\mathrm{DS}}$, and z_{DS}. These are the x-, y-, and z-displacements from the origin of the global coordinate system to the origin of the local coordinate system of the optical surface.
- Rotation specifications $\alpha_{\mathrm{RS}}, \beta_{\mathrm{RS}}$, and γ_{RS}. These are the angles in degrees by which the local coordinate system would need to be rotated in order to cause it to align with the global coordinate system. The rotations are specified as $-\gamma_{\mathrm{RS}}$ about the z-axis, β_{RS} about the y-axis, and α_{RS} about the x-axis.
- Rectangular aperture dimensions x_{RA} and y_{RA} (if the surface has a rectangular aperture). These are the half-width of the aperture in the x-direction and the half-height of the aperture in the y-direction. The aperture is regarded as lying in the xy-plane of the local coordinate system.
- Circular Aperture r_{CA} (if the surface has a circular aperture). This is the radius of the circular aperture. The aperture is regarded as lying in the xy-plane of the local coordinate system.
- Conic Constant k. The meaning of the possible values of k is given in Table I.

C. Image Plane

An element applicable to the image plane alone is as follows.

- Image plane curvature c_{IP}. For MODIS, $c_{\mathrm{IP}} = 0$ because the image plane is flat.

VI. 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 a Bittware Hammerhead 66-MHz PCI board containing four Analog Devices 21160 DSPs operating at 80 MHz. Our approach was to use the PC to direct the operation of the DSPs. All ray tracing

was performed using the mathematical models presented in Sections II and III. The the following are two main parts to the ray tracing program, one performed by the PC and the other performed by the DSPs.

PC Task) Trace 21 rows × 21 columns = 441 rays from each of 485 pinholes in the attenuating screen to the point where they strike the diffuser. There are 213 885 such points.

DSP Task) Trace 241 rows × 121 columns = 29 161 rays from each of these diffuser locations to where the rays strike the focal plane, if they do indeed do so. (If they do not pass through all apertures, then they fail to reach it.)

For the PC to dispatch DSP Tasks, we used C-language library routines provided by Bittware. These permit a C program running on a PC to download programs to each DSP individually. They also permit the PC program to modify and read the memory associated with each DSP.

To synchronize the DSPs with the PC, we implemented a full handshaking scheme. In this scheme, each DSP's memory contained two flags, **DSPReady** and **PCReady**. The sequence was as follows.

1) After a DSP finished a DSP Task, it needed to pass its results to the PC and obtain a new assignment. To do so it would set **DSPReady** ← **True** and idle while **PCReady** remained **False**.
2) When the PC discovered this (by polling), it would set **PCReady** ← **True**. At this point, it was free to retrieve the results of the last DSP Task (if any) and store parameters in the DSP's memory describing the next task. These parameters amounted to a point on the diffuser surface from which to start tracing rays.
3) Meanwhile, the DSP set **DSPReady** ← **False** and idled as long as **PCReady** was **True**.
4) On finishing its work, the PC would set **PCReady** ← **False**. When the DSP noticed this, it was free to start its next task and the PC would continue scanning for DSPs ready for new work assignments.

We put identical ray-tracing programs in all four DSPs on the Hammerhead board and performed tests using one, two, three, and all four of these.

Having determined a starting location for a DSP Task, the PC looked for an idle DSP. Upon finding one, it would retrieve the results from that DSP's last task and dispatch a new task to it. The image plane of the optical system was divided up into a rectangular grid of cells. The result of one DSP Task was a 2-D histogram showing the number of times a given cell was struck by all the rays in that task. The PC would accumulate the sum of all such histograms, yielding a composite indication of the number of rays striking any cell. It would then resume scanning of the available DSPs.

The overall ray-tracing scheme has not yet been fully completed. We have programmed the PC to assign tasks to the DSPs and the DSPs perform their tasks correctly. We consequently have been able to measure the time it takes to trace rays through the optical system. From this, we have been able to determine the increase of speed obtained by adding additional DSPs, as described below. However, the PC Task to determine all 213 885 rays striking the diffuser surface remains to be completed.

TABLE II
TIME TO TRACE 349 932 RAYS ON VARYING NUMBERS OF DSPS

Number of Processors	Elapsed Time (s)	Time per Ray (μs)	Expected Completion Time (days)
1	122.861	351.6	100.8
2	61.449	175.8	50.4
3	40.942	117.2	33.6
4	30.721	87.25	25.2

VII. PERFORMANCE

A. Baseline Performance

The algorithms described in this paper were implemented several years ago in FORTRAN and executed on a DEC Alpha computer. The program described in [8] performed ray tracing for 485 pinholes with roughly 441 rays emanating from each pinhole. Each of these rays generated a bundle of 29 161 rays reflected from the diffuser, for a grand total of roughly 6.24 G rays. It took the program around 14 days to complete the calculations, an average rate of 5.16 k raytraces/s, equivalent to roughly 194 μs/raytrace.

B. Performance on Parallel Processors

Table II gives performance measurements of the ray tracing algorithm when distributed to multiple DSPs executing in parallel. The results in the table were determined by repeatedly tracing the same ray 349 932 times. The ray selected was one which traversed all 28 surfaces of the MODIS optical system without missing any internal aperture. The number of times this ray was traced matched the number of rays in 12 complete bundles of rays leaving the same point of the solar diffuser, as explained in Section VI. In practice, numerous rays would miss some aperture, cutting short the time to trace such a ray, and so increasing the number of rays traced per unit time. The results shown here, therefore, are conservative. The time which would be required to complete all 6.24 G rays is shown in the rightmost column. Times were measured using the **clock**() function in the C runtime library and yielded times to the nearest millisecond.

Fig. 13 graphs the number of rays traced per second as a function of the number of processors used. A linear curve fit shows that within the range of one to four processors, the number of rays processed per second r is related to the number of processors n by

$$r = ((2847.97 \pm 1.1)n + (0.21 \pm 3.1)) \text{ rays} \cdot \text{s}^{-1} \qquad (46)$$

where the error bounds are ±1 standard deviation.

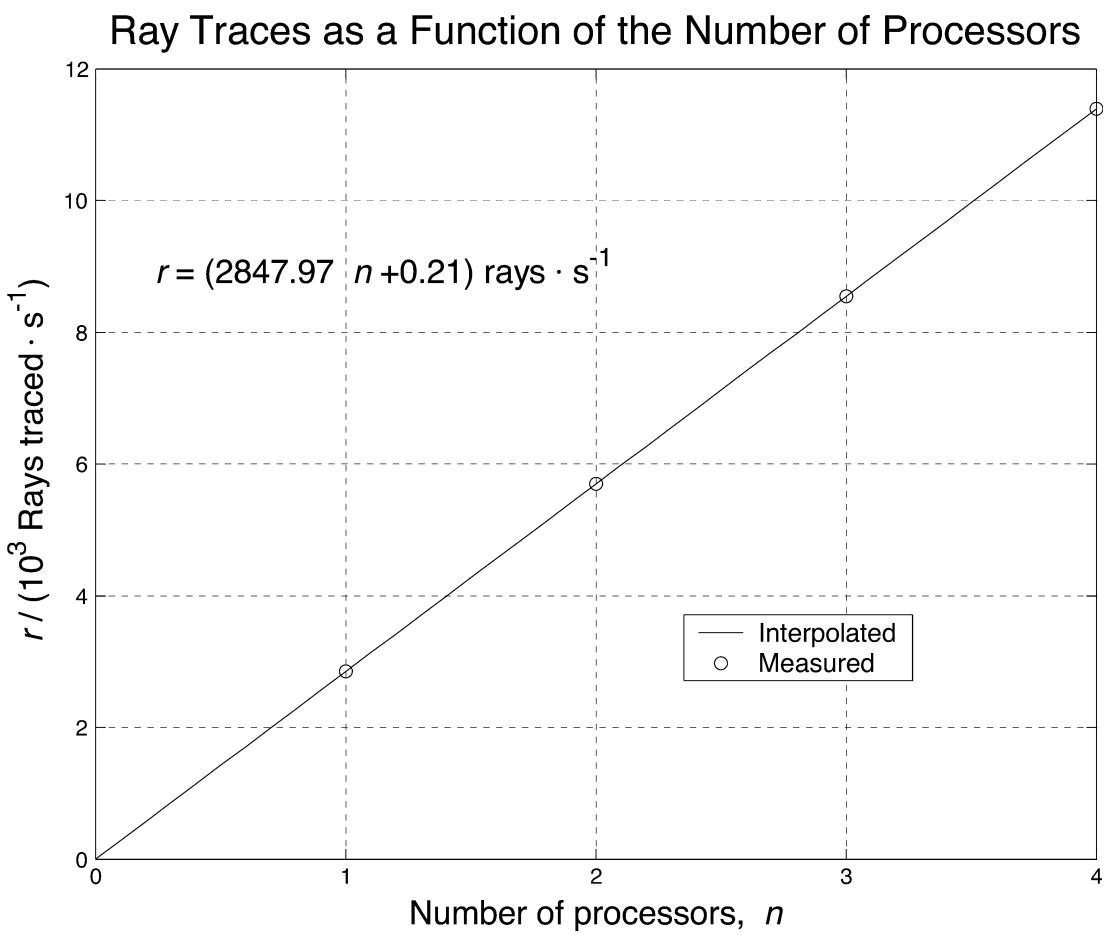

Fig. 13. Preliminary results of ray tracing on the DSP.

There are several additional pertinent observations to be made.

- It is apparent that there is a strongly linear relationship between the number of rays which can be traced in one second and the number of DSPs available to do the work. What is not so clear is how many processors could be handled in parallel before the host PC gets so busy processing results that it cannot keep up. Our measurements failed to show any delay for administrative tasks, although the time taken to trace a ray bundle of 29 161 rays on the DSPs, about 10 s per bundle, was easily measured. The negligible amount of overhead is reflected in the fact that the additive constant 0.21 ± 3.1 in (46) is very much smaller than the coefficient of n.

  Since the $\mathbf{clock}()$ routine can measure time to the nearest millisecond and the time for a single DSP to execute a bundle of work is around 10 000 ms, the ratio of productive time to administrative overhead appears to be at least 10 000:1. The implication is that the host PC could service up to 10 000 subordinate DSPs and still achieve a linearly increasing number of rays traced per unit time. With only four DSPs on each Hammerhead PCI board, that would represent 2 500 such boards, more than would ever fit on a PCI bus. Furthermore, it is likely that if more DSPs were in service, they would have to wait occasionally to get more work, whereas waiting time was minimal with just four DSPs. However, the very strong linear relationship between the number of DSPs used and the number of rays traced per second does demonstrate the significant performance improvement achievable by using parallel processors, implying that this approach is highly *scalable*.
- Even with four processors, the time it takes to complete the ray tracing exceeds that of the original uniprocessor solution. Using (46), we can see it would take 1.8 processors to match the baseline rate of 194 μs/raytrace.
- Table III shows extrapolations relating the number of DSPs applied and the cost of the necessary number of PCI boards (currently at \$5595 each) to the number of rays which could be traced in each second, along with the corresponding amount of time for tracing each ray.

  These estimates represent an upper limit. Many rays will not reach the detector and so tracing them will terminate early. Thus, the speeds achieved can be expected to exceed these estimates. These extrapolated estimates must of course be presumed to be ever less accurate as the number of DSPs rise.

  It would take two Hammerhead PCI boards, therefore, to exceed the performance of the original baseline program. At nearly \$12 000 for such a system, not counting

TABLE III
PROJECTION OF RAY TRACING SPEEDS WITH ADDITIONAL DSPs

# of DSPs	# of boards	Cost of boards ($)	Ray traces per second	Seconds per ray trace	Time to Completion (days)
10	3	16,785	28,480	35 μs	10.1
20	5	27,975	57,000	18 μs	5.0
50	13	72,735	142,000	7.0 μs	2.0

Bittware has a new board with eight 250 MHz ADSP-TS101 TigerSHARC digital signal processors and a Xilinx Virtex-II Pro FPGA. With an increase in processing speed by a factor of 6, this would permit marked reductions in these projected completion times.

the PC containing it, this is an expensive approach. However, a combination of optimizing the code and obtaining even faster results by just adding more DSPs makes it worth exploring further. Also, the baseline performance includes numerous rays which never reach the focal plane, whereas the experimental results reported here include only rays which do make the entire journey.

- The use of the AD21160 for this application is not optimal. This DSP has been optimized for typical signal processing applications in which multiplications and additions are prevalent. In contrast, our ray tracing algorithm also requires repeated computation of reciprocals and square roots. The AD21160 does not provide hardware reciprocal or square root computations. It does have instructions which give an initial "guess" at reciprocals and square roots but additional software instructions are required to refine these guesses. Two alternative means which could be used to speed up the processing are either to choose a different kind of subordinate processor with more powerful floating-point capabilities or offload those computations from the DSP to a field programmable gate-array device. Of course, there are some offsetting advantages to the use of DSPs: they do not dissipate as much heat as processors such as the Intel Pentium IV, and harnessing them to work in parallel is quite easy.
- There has been very little effort expended so far on optimizing the algorithm executed in the DSPs. Doing so would have the effect of reducing the execution time. For example, it might be possible to reduce the use of the operations of division and square-root extraction (which are lengthy on the AD21160 DSPs). For example, the square root operation entailed in (15) is avoided if we use the approximation in (3). We are now exploring a mesh solution to obtain the intersection of a ray with a conicoid more efficiently.
- The problem of tracing rays in a system such as MODIS is dominated by computations as opposed to input/output operations.) This implies that the computation could be distributed to PCs connected to the Internet because the demands of communications would be low compared to the amount of calculation required. This is the method being used in the Great Mersenne Prime Search,[6] where numerous volunteers around the world permit idle time on their computers to be used to search for the elusive Mersenne primes (only 41 have ever been discovered.) On the one hand, PCs doing other work simultaneously would have an adverse effect on the overall completion rate. On the other hand, devoting PCs solely to the ray-tracing task would be a rather expensive approach since PCs are general-purpose computers and special-purpose computers are adequate to the task.

VIII. CONCLUSION

Standard commercial ray-tracing systems are not well suited to use with an optical system like that in MODIS because of its employment of an attenuator consisting of a grid of pinholes, each of which acts like a separate light source. Achieving a full understanding of the combined effect of such an attenuator can be achieved by the use of a computer simulation to trace individual light rays to their destination in the focal plane, if indeed they actually do reach it. We implemented such a simulator using a small array of Analog Devices AD21160 digital signal processing microprocessors programmed in C. They were under the control of a program also written in C and executing on a Dell PC using the Windows 2000 operating system.

We divided the ray tracing task into two parts. The first of these, performed by the PC, entailed locating the spots on the diffuser plane reached by individual rays from the sun. Subsequently, the cone of rays departing each such spot was traced by an assigned DSP. A 2-D histogram representing the number of rays striking each location in the focal plane can be accumulated by the PC program, permitting determination of the overall illumination pattern on the screen without actually building an instrument. This, in turn, permits the exploration of different configurations of holes on an attenuator screen in an attempt to create a more uniform distribution of light intensities in the focal plane.

Preliminary experiments show a very highly linear relationship between the speed of such processing and the number of processors in the system. Tradeoffs between cost and speed can now be considered. Further work entails the following nonexhaustive list of tasks:

- completing the program so that all parts of the simulation can be performed;
- improving the efficiency of the ray tracing algorithms by optimizing loops and by minimizing the use of division and square root operations;
- considering the use of alternate processors with more efficient floating point capabilities (for example, the 600-MHz ADSP-TS201S TigerSHARC);
- exploring the use of processors distributed widely across a network;
- investigating the limitations of expansion using the PCI bus; and

[6]Mersenne Prime Search (2003). http://www.mersenne.org/prime.htm

- measuring the speed of execution of the algorithm on PCs of various kinds.

APPENDIX
NOTATION

A scalar value x is represented using italic font.

A point $\mathsf{P} = (x, y, z)$ is represented in sans-serif font.

A unit-length vector $\hat{\boldsymbol{a}}$ is represented in bold italic with a circumflex above. The unit vectors parallel to the x-, y-, and z-axes are $\hat{\boldsymbol{u}}_x, \hat{\boldsymbol{u}}_y$, and $\hat{\boldsymbol{u}}_z$, respectively.

A vector $\boldsymbol{r} = r_x\hat{\boldsymbol{u}}_x + r_y\hat{\boldsymbol{u}}_y + r_z\hat{\boldsymbol{u}}_z$ is represented in bold italic. We frequently treat a point P as if it were equivalent to its vector representation $\boldsymbol{P}$.

The dot product of two vectors is a scalar quantity defined by $\boldsymbol{r}_1 \cdot \boldsymbol{r}_2 = |\boldsymbol{r}_1||\boldsymbol{r}_2| \cos\theta$, where θ is the angle separating the two vectors.

The cross product of two vectors is a vector whose magnitude is given by $\boldsymbol{r}_1 \times \boldsymbol{r}_2 = |\boldsymbol{r}_1||\boldsymbol{r}_2| \sin\theta$, where θ is the angle separating the two vectors. The direction of the cross product is determined by the right-hand rule. In Cartesian coordinates the cross product can be computed as the determinant

$$\boldsymbol{r}_1 \times \boldsymbol{r}_2 = \begin{vmatrix} \hat{\boldsymbol{u}}_x & \hat{\boldsymbol{u}}_y & \hat{\boldsymbol{u}}_z \\ r_{1x} & r_{1y} & r_{1z} \\ r_{2x} & r_{2y} & r_{2z} \end{vmatrix}. \tag{47}$$

We represent the direction of a ray by a unit-vector parallel to it. For example, $\hat{\boldsymbol{\omega}} = L\hat{\boldsymbol{u}}_x + M\hat{\boldsymbol{u}}_y + N\hat{\boldsymbol{u}}_z = \cos\theta_x\hat{\boldsymbol{u}}_x + \cos\theta_y\hat{\boldsymbol{u}}_y + \cos\theta_z\hat{\boldsymbol{u}}_z$. L, M, and N are the direction cosines of the ray and $L^2 + M^2 + N^2 = 1$.

REFERENCES

[1] E. Hecht, *Optics*. Reading, MA: Addison-Wesley, May 1987, corrected 1990.

[2] G. H. Spencer and M. V. R. K. Murtry, "General ray-tracing procedure," *J. Opt. Soc. Ameri.*, vol. 52, no. 6, pp. 652–678, June 1962.

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

[4] W. T. Welford, *Aberrations of Optical Systems*. New York: Adam Hilger, 1986.

[5] W. J. Smith, *Modern Optical Engineering*, 3rd ed. New York: McGraw-Hill, 2000.

[6] A. Nussbaum, *Optical System Design*. Upper Saddle River, NJ: Prentice-Hall, 1998.

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

[8] E. Waluschka, J. Esposito, J. Sun, X. Wang, and X. Xiong, "MODIS solar diffuser—modeled and actual performance," in *Proc. SPIE Earth Observing Syst. VI*, vol. 4483, W. L. Barnes, Ed., 2001, pp. 146–155.

[9] NASA Goddard Space Flight Center sun fact sheet (2002, June). [Online]. Available: http://nssdc.gsfc.nasa.gov/planetary/factsheet/sunfact.html

[10] S. Kizhner, D. J. Petrick, T. P. Flatley, P. Hestnes, M. Jentoft-Nilsen, and K. Blank, "Pre-hardware optimization of spacecraft image processing software algorithms and hardware implementation," in *Proc. IEEE Aerospace Conf.*, vol. 4, Mar., 9–16 2002, pp. 1975–1992. IEEE Catalog Number: 02TH8593C.

[11] *ADSP-21160 SHARC DSP Hardware Reference*, 1st ed., part Number 82-001 966-01, Analog Devices, Inc., Norwood, MA, Nov. 1999.

[12] *DspHost Library Reference Manual*, release 6.3, Bittware, Inc., Concord, New Hampshire, Aug. 6, 2001.

[13] W. H. Beyer, Ed., *CRC Standard Mathematical Tables*, 26th ed. Boca Raton, FL: CRC, 1981.

Charles B. Cameron (M'91) received the B.Sc. degree in computer science from the University of Toronto, ON, CA, in 1977 and the M.S.E.E. and Ph.D. degrees from the Naval Postgraduate School, Monterey, CA, in 1989 and 1991, respectively.

A Commander in the United States 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 fiber-optic sensors operating as interferometers.

Rosa Nívea Rodríguez (M'02) was born in Puerto Rico in l982. She received the B.S. degree in computer engineering in May 2004 from the Universidad de Puerto Rico (Mayagüez).

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

Ms. Nívea Rodríguez was the President of the Computer Society student chapter at the Mayagüez Campus of the Universidad de Puerto Rico, a member of the Society of Women Engineers (SWE), and a member of the Tau Beta Pi Honor Society.

Nathan Padgett was born in Washington, DC, in 1982. He is currently pursuing the B.Sc. degree in computer science at the Georgia Intsitute of Technology. He also is in the Air Force Reserve Officer Training Corps (ROTC) and is slated to attend undergraduate pilot training upon graduation in May 2005.

His current research interests include 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.

Eugene Waluschka received the Ph.D. degree in physics from New York University, New York, 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, Norwalk, CT, where most of his time was spent in modeling and numerically simulating various phenomena. Currently, he is with the NASA Goddard Space Flight Center, 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 the Johns Hopkins University, Baltimore, MD.

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.