JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 33, No. 1, January–February 2010

Moon-Tracking Modes for Star Trackers

John Enright*
Ryerson University, Toronto, Ontario M5B 2K3, Canada

DOI: 10.2514/1.42716

We examine the effectiveness and robustness of a moon-tracking backup mode for modern star trackers. This approach can be used to calculate a three-axis attitude solution from overexposed images of the moon. In examining this problem, we carefully consider the operational conditions under which this capability would be necessary and tailor our algorithmic approach appropriately. We describe a detailed simulation of the moon imaging process and outline the required image processing steps required to extract useful information from the moon images. The image processing extracts the moon outline and estimates the moon position and orientation using a nonlinear least-squares fit. Under most lighting conditions, our moon-vector estimates have an error of about 0.001 deg, and our sun-vector estimates have an error less than 0.25 deg. The capability of maintaining moderate accuracy attitude tracking in the presence of moon incursions is an important milestone toward a star-tracker-only attitude control system for microsatellites and nanosatellites.

Nomenclature

A, B, C, D	=	pixel vertex points
a	=	radius of moon image, pixels
b	=	semiminor axis of terminator curves, pixels
$\mathbf{C}_{FG}$	=	rotation matrix (from G to F)
E	=	overexposure factor
e	=	eccentricity
f_i	=	error function
f_{pix}	=	focal length, pixels
$\mathbf{g}$	=	curve-fitting error
$\mathbf{h}_G$	=	focus blur impulse response
I_{thresh}	=	image intensity threshold
$\mathbf{J}$	=	Jacobian matrix
k	=	illuminated fraction
M_c	=	number of points on circular (limb) curve
M_e	=	number of points on elliptical (terminator) curve
M_f	=	number of fitting parameters
M_s	=	number of perimeter points
m, n	=	image indices
N_G	=	size of focus blur filter
N_r, N_c	=	number of rows/columns
N_s	=	number of perimeter pixels
N_{trial}	=	number of images per trial
P	=	phase law (visible magnitude)
$\mathbf{p}$	=	coordinates of pixel center
r_{EM}	=	mean Earth–moon orbital radius
r_{moon}	=	moon radius
$\hat{\mathbf{s}}_{MS}$	=	moon–sun unit vector
$\hat{\mathbf{s}}_{TM}$	=	spacecraft–moon unit vector
t	=	time
$\mathbf{u}$	=	parameter vector
V	=	visual magnitude
$\mathbf{x}$	=	column vector
x, y	=	image coordinates
x_c, y_c	=	moon centroid
x_0, y_0	=	center of moon disk
α	=	moon rotation angle
$\boldsymbol{\beta}$	=	moon image
$\boldsymbol{\beta}_B$	=	binary moon image
$\boldsymbol{\beta}_T$	=	thresholded moon image
Δ_m	=	error in $\hat{s}_{cm}$, rad
Δ_α	=	error in $\tilde{\alpha}$ (single image), rad
δ	=	pixel size, m
ξ	=	lunar phase angle
σ_G	=	width of focus blur, pixels
σ_m	=	standard deviation in $\hat{s}_{TM}$
σ_s	=	standard deviation in $\hat{s}_{MS}$
Φ	=	stellar flux
φ_i	=	angular coordinate of fitting point
Ψ	=	least-squares cost function
$\hat{(\cdot)}$	=	unit vector

I. Introduction

MODERN star trackers are an enabling technology for microsatellites and nanosatellites. This class of mission frequently demands a robust, moderate accuracy, three-axis attitude solution. A single low-cost star tracker can potentially simplify spacecraft design by avoiding the need for multiple sensors (e.g., sun sensors, horizon sensor, and magnetometers), but significant barriers exist to adopting star-tracker-only attitude estimation systems. One situation that can affect the robustness of star trackers are moon incursions into the sensor field of view. This paper proposes a detailed strategy for using star-tracker-derived moon images to obtain a three-axis attitude fix. This alternate operating mode can be used when stray light from the moon precludes the use of standard star-tracking routines.

The moon position in an image lets us estimate the direction to the moon in body coordinates; the shape of the partially illuminated moon allows us to calculate the direction vector to the sun. These two measurements are sufficient to estimate the inertial attitude of the satellite. Our algorithms begin by adjusting the star-tracker integration time to deliberately overexpose the moon image. This strategy removes visible features on the surface of the moon and improves the contrast between the moon disk and surrounding space. The overexposed moon images have a crisp boundary curve without the need for dilation and erosion processing steps. Our algorithms then identify the boundary pixels and use a nonlinear least-squares fit to match the observed shape to a parametric model of the moon. Algorithm performance is verified by a detailed simulation. The inputs to this simulation are based on a prototype star-tracker design currently in development.

Other researchers have investigated this problem in the past. Mortari has been the primary proponent of a moon–sun attitude sensor [1]. In his work, he and his colleagues have proposed a dedicated attitude sensor that uses images of the moon to determine

Received 12 December 2008; revision received 20 August 2009; accepted for publication 6 September 2009.

*Associate Professor, Department of Aerospace Engineering, 350 Victoria Street. Member AIAA.

Table 1 Frames of reference

Frame	Notation	Origin	x axis	y axis	z axis
		Space reference frames			
Inertial	I	Optical axis and projected focal plane	Arbitrary[a]	Arbitrary	Arbitrary
Satellite-sun-moon	R	Optical axis and projected focal plane	Projection of $\hat{s}_{MS}$ onto plane normal to $\hat{s}_{TM}$	——	$\hat{s}_{TM}$
Star tracker	T	Optical axis and projected focal plane	Detector columns	Detector rows	Boresight (outward)
		Detector reference frames			
Image	F	Corner pixel of detector projected focal plane	Detector columns	Detector rows	——
Moon centroid	G	Center of moon image	Detector columns	Detector rows	——
Moon limb	L	Center of moon image	Moon limb	Line of cusps	——

[a]Any convenient inertial frame such as the Earth-centered inertial will suffice.

satellite attitude. This device takes high-resolution images of the moon and then performs least-squares or correlation-based [2] registration to a reference moon image. Their attitude solution is applicable to different systems, but their method for image analysis differs substantively from our approach. Pingyuan et al. have proposed an Earth–sun sensor for use by lunar rovers [3]. Their image processing approach is similar to ours: edge identification followed by nonlinear least-squares fitting, but their model formulation and target application differs.

Throughout this paper, vector quantities are denoted by boldface, lowercase letters, for example, $\mathbf{x}$. Unit vectors are denoted with an additional caret, for example, $\hat{\mathbf{u}}$. When vectors are expressed in matrix form relative to a particular frame of reference, they are subscripted with a capital letter denoting the reference frame. Thus, $\mathbf{x}_F$ and $\mathbf{x}_G$ are two matrix representations of the same abstract vector, $\mathbf{x}$, expressed in frames F and G, respectively. The orthogonal basis vectors that define our frames of reference are denoted similarly; for instance, $\hat{\mathbf{F}}_x$, $\hat{\mathbf{F}}_y$, and $\hat{\mathbf{F}}_z$ are the basis vectors of F.

Several frames of reference are used throughout this paper (Table 1). Three space-referenced frames are used to determine the attitude solution from the instrument observations: the star-tracker frame is fixed to the spacecraft body, the satellite–moon–sun frame is defined relative to those three bodies, and the inertial frame can be any fixed external frame. In addition to these three-dimensional frames of reference, much of the image analysis takes place in the plane of the detector. We treat vectors in the three detector frames as two-element vectors (with the out-of-plane components implicitly zero). Relationships between the detector frames and the space frames are derived as needed.

The remainder of this paper is organized as follows. Section II describes the theoretical background and operational motivation for this study. Section III introduces the relations necessary to accurately simulate overexposed images of the moon. Our moon-tracking algorithms are developed in Sec. IV and tested in various mathematical configurations in Sec. V. The conclusions in Sec. VI summarize the contributions of this paper and provide suggestions for future work.

II. Partially Illuminated Moon

In this section, we consider several background topics that motivate this research and contribute to the analysis presented in the paper. First, we describe the geometry of moon images and review the nomenclature necessary to discuss notable image features. Second, we sketch out the design of a prototype star tracker. This design provides the baseline values for the image parameters used in the rest of this paper. Third, we reinforce the motivation for this work by presenting a more detailed examination of the effects of stray light on star-tracker performance. The section concludes with a review of how satellite attitude can be calculated from the moon images.

A. Appearance of the Moon

Ignoring surface features, the outline of the partially illuminated moon can be described by two curves: a half-circle formed by the edge of the moon disk, and a half-ellipse formed by the terminator [4]. The elliptical portion of a crescent moon (i.e., less than half-full) forms a concave shape, while in a gibbous moon the image shape is convex. The semicircle and semi-ellipse curves share a common axis, and meet at the cusps (Fig. 1). The extent of the illuminated portion of the moon is characterized by the quantity k, which can be interpreted as either a linear ratio

$$k = AB/AC \tag{1}$$

or a ratio of the illuminated area to the total area of the moon disk. Because the moon, the sun, and the satellite define a plane, we can relate the illuminated fraction k to the phase angle ξ, the angle between the observer and the sun, as viewed from the moon (Fig. 2). This figure also illustrates several of the frames of reference introduced in Table 1. The relationship between these two quantities can be derived from planar geometry:

$$k = \frac{1 + \cos\xi}{2} \tag{2}$$

The geometry of an arbitrary moon image can be reduced to a few simple parameters (Fig. 3). The center of the moon disk is located at a position (x_0, y_0) in the image. This is usually expressed in units of pixels, in the row and column directions, respectively. The line of cusps, and hence the major axis of the ellipse, is rotated by an angle α. This rotation defines the translated and rotated frame of reference L. The semicircle of the moon disk has radius a. The rotation angle α is chosen such that the semicircle always corresponds to the points where $y_L \geq 0$. The elliptical boundary has semimajor axis a and semiminor axis b. Conventionally, b is always positive, however, if we allow $-a \leq b \leq a$, then there is a convenient one-to-one mapping between k and b, namely,

$$b = a \cdot (1 - 2k) \tag{3}$$

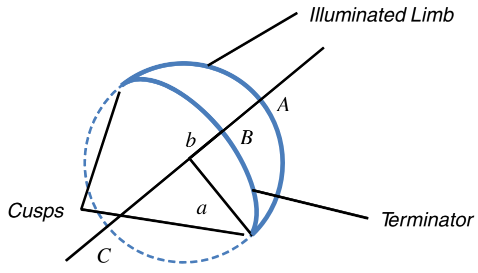

Fig. 1 Geometry of the partially illuminated moon.

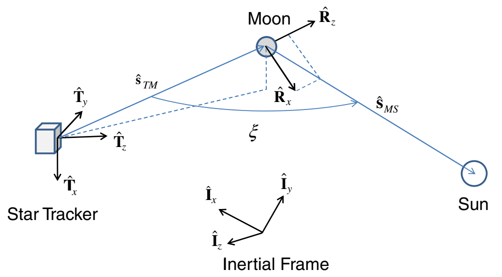

Fig. 2 Moon–sun–satellite geometry.

Thus, a positive b corresponds to a crescent moon, and a negative b corresponds to a gibbous moon.

B. Baseline Star-Tracker Concept

This paper represents a companion study to an ongoing effort to develop a very small star tracker using high-quality commercial parts. Component selection for our star-tracker design establishes baseline parameters relevant to the moon-tracking problem. Our chosen detector is a 5 megapixel (2592×1944 pixel), 7.13 mm (diag.) complementary metal oxide semiconductor (CMOS) sensor. In keeping with our aim of minimizing mass and volume, the optics are very compact; the current design employs a 6 mm (diameter), $f2.0$ lens. The resulting full-angle field of view (FOV) from this design is approximately 26×20 deg. This FOV is comparable to many commercially available wide-angle star trackers, but the native resolution is higher than the current norm. This sensor is designed to track stars of magnitude 4.0 or brighter.

Our prototype sensor design represents the target platform for our moon-tracking approach, but we would like to maintain generality in our analysis. We observe that many of the details of the optical design can be abstracted away if a is expressed in units of pixels. Adopting this approach captures the primary effects of the detector pixel pitch, detector size, and optical focal length. Although the angular radius of the moon will change as the satellite orbits, even geostationary satellites will only see about a 10% variation in the moon's size. Most small spacecraft use low Earth orbits, where the variations are much smaller. Thus, the choice of optical components will have the greatest effect on a. For Earth-based observations, the baseline design gives $a \approx 25$ pixels.

Although similar in design to a camera, several factors distinguish a star tracker from a conventional imager. First, it is a common practice in star-tracker design to deliberately defocus the optics. Spreading the incident starlight over a number of pixels makes it easier to accurately measure the locations of star centroids. Thus, images of the moon will be blurry and may appear larger than we would normally expect. Second, we rely on sufficiently long exposure times to saturate the images and remove visible surface features. A judicious choice of the moon exposure time is necessary to ensure good quality images. A composite of several moon images with varying exposure times is shown in Fig. 4. These images were taken with our prototype star tracker and help to guide the image simulation sections of this study. Short exposure times yield faint images with poorly defined edges and visible surface features. As exposure times get very long, the distinct shape of the moon is lost.

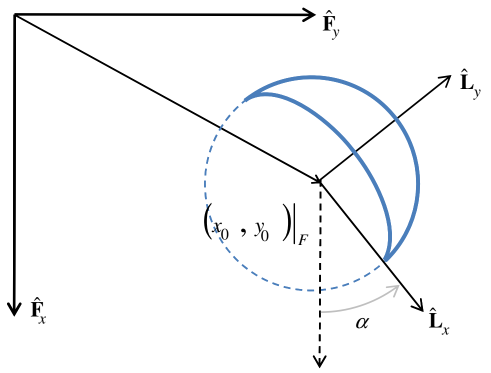

Fig. 3 Parametrization of an arbitrary moon image.

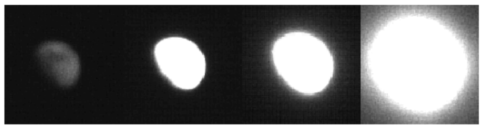

Fig. 4 Moon images from prototype star tracker. Exposure times from left to right: 100 μs, 1 ms, 10 ms, 100 ms.

For this set of images, the 1 ms exposure is ideal: the moon shape is well defined and visible artifacts are nearly absent. By comparison, the typical star-tracking exposure time for this sensor is 100 ms, and so we expect to see very little response from starlight when operating in moon-tracking mode. We also note that the apparent radius of the moon in this image has grown somewhat; here, $a \approx 30$ pixels due to the blurring and exposure effects (by way of comparison, stars have a radius of about 2–3 pixels). We adopt this new value of a as the default moon radius and assume that exposure time on the star tracker can be adjusted dynamically to maintain this type of image.

C. Stray Light

The extent to which light from the moon will affect nominal star tracking depends greatly on the specific design of the star tracker. Essentially, problems arise when there is sufficient light from the moon reaching the detector to overwhelm the natural response to the target stars. Various physical processes contribute to this effect [5]. When the moon is in or near the field of view, light may diffract around or reflect off the knife edges found on the instrument baffling. Alternately, internal scattering within the optics and sensor housing may direct light onto parts of the array that would otherwise remain unilluminated. Finally, some detector technologies may be subject to blooming, an electrical effect caused by overfilled detector wells. Regardless of its origin, stray light can reduce a sensor's star tracking effectiveness. Quantitative predictions of the effect of stray moon-light is difficult and is heavily dependent on sensor and mission design. Some mission operations' reports have reported very little deterioration in tracking availability [6], while others seem fairly sensitive [7].

Although absolute predictions of moon effects are difficult, we can assess bounds on the operational impact of stray light by considering the visual magnitude of the moon compared to the magnitude of a sensor's target stars. The apparent visual magnitudes of two objects, V_1, V_2, can be related to their incident fluxes, Φ_1, Φ_2, through

$$\Delta V = V_2 - V_1 = \frac{5}{2} \cdot \log_{10}\left(\frac{\Phi_1}{\Phi_2}\right) \tag{4}$$

Thus, every order of magnitude difference in flux intensity corresponds to a difference in visual magnitude of 2.5 (positive numbers are dimmer). Researchers have reported [5] that typical baffle construction can reduce the incident flux by a factors of 10^{-5}–10^{-6} when a bright object is near or within the FOV. Thus, when pointed near the moon, we expect to see some stray-light response with a visual magnitude of 12.5–15 dimmer than the moon:

$$V_{\text{stray}} = V_{\text{moon}} + \Delta V \tag{5}$$

The apparent visual magnitude of the moon is dependent on its phase and the location of the observer. Empirical study [8] has shown that the flux from the moon is not simply scaled by k, but follows its own phase law $P(\xi)$. Tabulated values for this function are provided in Table 2. Intermediate values are obtained with spline interpolation.

For an observer at a distance R (measured in AU), from the moon [8], gives the apparent magnitude as

Table 2 Lunar phase law (reproduced from [8])

ξ, deg	$P(\xi)$	ξ, deg	$P(\xi)$
0	1.000	80	0.127
5	0.929	90	0.089
10	0.809	100	0.061
20	0.625	110	0.041
30	0.483	120	0.027
40	0.377	130	0.017
50	0.288	140	0.009
60	0.225	150	0.004
70	0.172	160	0.001

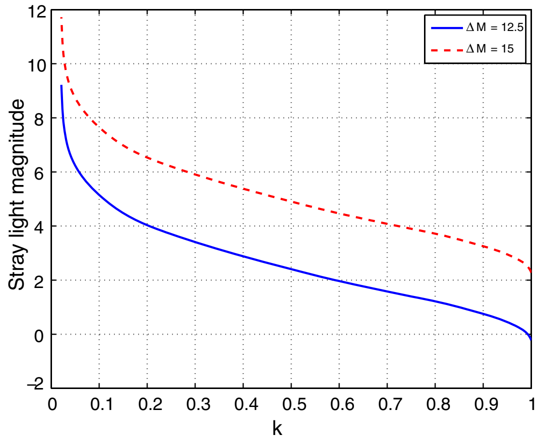

Fig. 5 Stray-light magnitude from the moon.

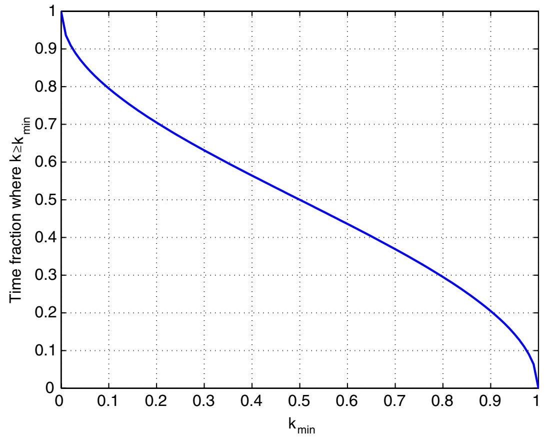

Fig. 6 Time distribution of *k*. The dependent axis represents the fraction of time with *k* greater than the specified value.

$$V_{\text{moon}} = 0.23 + 5 \cdot \log_{10} R - 2.5 \cdot \log_{10} P(\xi) \tag{6}$$

Using Eqs. (2) and (6) we can then calculate the apparent magnitude of the moon for a satellite in orbit near the Earth (i.e., $R = R_{em}$). This is shown in Fig. 5 for a few values of ΔV. This type of plot can be used to help establish a lower operational bound on when moon tracking is needed. For instance, if our star tracker relies on magnitude 4.0 stars, and we wish to have an order of magnitude difference between stray light and our target stars, then the moon-tracking mode must operate when the stray-light magnitude is 6.5. If the baffle scattering is relatively large, that is, $\Delta V = 12.5$, this corresponds to $k_{\min} \approx 0.04$, a very small crescent. If, however, $\Delta V = 15$, then we meet this performance when $k_{\min} \approx 0.2$. Not only does this determine the range of images that must be handled by the algorithms, but it also provides a sense of the fraction of time that the system can operate.

If we wish to find an expression for $t_{\text{frac}}(k_{\min})$, the fraction of time with $k \geq k_{\min}$, we must revisit the satellite–moon–sun system (Fig. 2) and consider how the geometry changes with time. With an eccentricity of 0.05, the moon's orbit is fairly circular. Consequently, we may assume $\mathrm{d}\xi/\mathrm{d}t \approx \text{constant} = 2\pi(\text{rad}/\text{lunation})^{\dagger}$. Thus,

$$\mathrm{d}\xi = 2\pi \mathrm{d}t \tag{7}$$

The integration of this expression is trivial, but some care must be taken in choosing the limits of integration. To help us find $t_{\text{frac}}\,(k_{\min})$, we choose the limits so that the integration spans the time between $k = k_{\min}$ and the full moon. Thus, we can get

$$\xi|_{k=1} - \xi|_{k=k_{\min}} = 2\pi(t_{\text{full}} - t_{k_{\min}}) \tag{8}$$

†A lunation is another term for the mean synodic period of the moon.

From Eq. (2), we can rewrite this expression in terms of k:

$$\arccos(2k_{\min} - 1) = 2\pi(t_{k_{\min}} - t_{\text{full}}) \tag{9}$$

Thus, solving for the time spanned by the integration and noting that arccos is even gives

$$(t_{\text{full}} - t_{k_{\min}}) = \frac{1}{2\pi}\arccos(2k_{\min} - 1) \tag{10}$$

The waxing and waning phases of the moon's orbit are symmetric, and so the total span of time with $k \geq k_{\min}$ is twice this amount. We also note that, because the time unit chosen for this analysis is a lunation, the left-hand side of this expression is effectively a time fraction. Thus,

$$t_{\text{frac}}(k_{\min}) = \frac{1}{\pi}\arccos(2k_{\min} - 1) \tag{11}$$

A plot of this function is shown in Fig. 6. Because of the cosine term in Eq. (2), the moon spends a fair amount of time near the extremes of k. For example, more than 25% of the time, k is outside of the range of $0.04 \leq k < 0.96$. This is relevant when discussing the operational requirements of the moon-tracking mode (see Sec. V for a more complete discussion of algorithm performance in these regimes).

D. Satellite Attitude Solutions

In this section, we provide a short overview of the relationships between image features and observation vectors. Mortari [1] provides a more comprehensive treatment of this problem. The three-axis attitude solution from the moon-tracking system relies on measuring two direction vectors relative to the sensor: $\hat{s}_{TM}$, the unit vector from spacecraft to moon, and $\hat{s}_{MS}$, the unit vector from moon to the sun. Both vectors are measured in T and are compared to ephemeris predictions of the same vectors made in I. Figure 7 illustrates the correspondence between image features and three-dimensional direction vectors. In this figure, we use the projected focal plane; this approach avoids the image inversion caused by the optics.

The spacecraft–moon vector can be obtained from the location of the moon center $(x_o, y_0)_T$ in the detector image. If the lens system has a focal length f_{pix} expressed in pixels, then the corresponding boresight angle to the actual moon center is

$$\rho_{TM} = \arctan\left(\frac{\sqrt{x_0^2 + y_0^2}}{f_{\text{pix}}}\right) \tag{12}$$

and the direction relative to the camera's x axis is

$$\theta_{TM} = \arctan 2(y_0, x_0) \tag{13}$$

Thus, expressed in the star-tracker frame

$$\hat{\mathbf{s}}_{TM_T} = \begin{bmatrix} \cos\theta_{TM} \sin\rho_{TM} \\ \sin\theta_{TM} \sin\rho_{TM} \\ \cos\rho_{TM} \end{bmatrix} \tag{14}$$

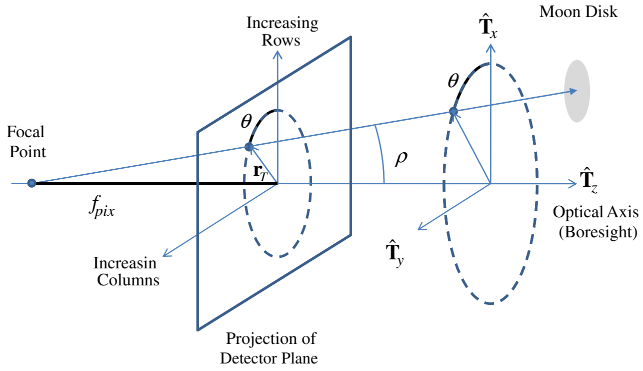

Fig. 7 Focal plane and sensor field of view.

To find the moon–sun vector, we must first express $\hat{\mathbf{s}}_{MS}$ in frame R and then find the transformation between R and T. From Fig. 2, we can see that $\hat{\mathbf{s}}_{MS_R}$ is a function of phase angle ξ:

$$\begin{aligned}\hat{\mathbf{s}}_{MS_R} &= \sin(\pi-\xi)\cdot\hat{\mathbf{R}}_x + \cos(\pi-\xi)\cdot\hat{\mathbf{R}}_z \\ &= \sin\xi\cdot\hat{\mathbf{R}}_x - \cos\xi\cdot\hat{\mathbf{R}}_z\end{aligned} \tag{15}$$

To find the transformation into frame T, we must express the basis vectors of R in the star-tracker frame. By the definition of these frames it is clear that

$$\hat{\mathbf{R}}_{z_T} \equiv \hat{\mathbf{s}}_{TM_T} \tag{16}$$

The projections of the moon–sun vector into the detector plane is a function of the rotation angle α. Recall that the x-axis basis vector of the moon-limb frame $\hat{\mathbf{L}}_x$ points from the center of the moon disk to the moon limb. This vector can be expressed in the star-tracker frame:

$$\hat{\mathbf{L}}_{x_T} = \begin{bmatrix}\cos\alpha \\ \sin\alpha \\ 0\end{bmatrix} \tag{17}$$

The vectors $\hat{L}_x$ and $\hat{R}_x$ must be coplanar, thus we can find a convenient expression for $\hat{R}_{x_T}$:

$$\hat{\mathbf{R}}_{x_T} = \frac{\hat{\mathbf{R}}_{z_T}\times(\hat{\mathbf{L}}_{x_T}\times\hat{\mathbf{R}}_{z_T})}{\|\hat{\mathbf{R}}_{z_T}\times(\hat{\mathbf{L}}_{x_T}\times\hat{\mathbf{R}}_{z_T})\|} \tag{18}$$

The third basis vector of R is chosen to preserve the right-handedness of the reference frame, and the transformation matrix from R to T is then simply

$$\mathbf{C}_{TR} \equiv [\,\hat{\mathbf{R}}_{x_T} \quad \hat{\mathbf{R}}_{y_T} \quad \hat{\mathbf{R}}_{z_T}\,] \tag{19}$$

We obtain an inertial attitude solution by comparing the measurement vectors in the star-tracker frame to their counterparts in an appropriate external frame. The two truth vectors $\hat{\mathbf{s}}_{ms_I}$ and $\hat{\mathbf{s}}_{cm_I}$ are found through the calculated ephemerides of the sun, moon, and satellite. Moderate accuracy ephemerides for the sun and moon are well understood (e.g., see Meeus [4]); the satellite position can be supplied by an orbital propagator or measured by onboard instruments. Once all four unit vectors are available, the attitude can be calculated with any appropriate technique (Crassidis et al. [9] provides a survey of contemporary methods).

III. Generating Synthetic Images

True moon images are useful for system validation, but they are impractical for general studies. To evaluate the performance of moon-tracking modes for an arbitrary optical design, we must be able to generate synthetic moon images in arbitrary orientations, positions, and illumination conditions. Throughout this derivation, we make several important assumptions. First, we neglect the case of bright stars near the moon disk. Although occultations between bright stars and the dim crescent moon do occur, they are infrequent. An outlier rejection algorithm could be used in a deployed system to resolve any ambiguity between moon and star image components. Second, we consider only monochrome images, an assumption consistent with contemporary star-tracker designs. Third, we assume that the moon disk remains overexposed and hence does not display any surface features. The resulting images are similar to real images from our prototype star tracker. Additionally, we consider only moon images that are wholly within the sensor field of view. Proper handling of moon images on the image boundary is a special topic reserved for future study.

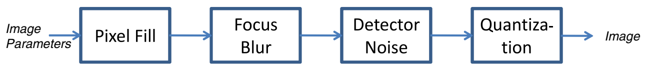

Fig. 8 Simulating moon images.

Simulating realistic moon images can be broken down into several processing steps (Fig. 8). We begin with a set of parameters that describe the moon position, orientation, and illumination. These parameters are used to calculate the fractional illumination of each pixel. The image is processed with a Gaussian filter to represent the effects of the optical defocus. Detector noise is then added to each pixel. Finally, the resulting image is quantized and thresholded to capture the effects of exposure time and finite dynamic range.

In this analysis, images can be considered two-dimensional matrices of pixel intensity values, for example, $\boldsymbol{\beta}$. Where necessary, we can refer to individual elements using subscripted row and column indices, that is, $\boldsymbol{\beta}_{m,n}$. Most image operations are performed on an elementwise basis, and so where the usage is unambiguous, index subscripts are omitted for notational clarity.

A simulated image $\boldsymbol{\beta}$ has N_r rows and N_c columns. The images are generated such that

$$N_r = N_c = \lceil 2.6\cdot a\rceil \tag{20}$$

That is, the image is 30% larger than the moon diameter. For most sensors, this would not represent the full extent of a star-tracker detector. Rather, it represents a coarse window around the moon and allows us to vary the moon position in the image.

A. Pixel Fill

The primary quantities used to construct the sensor images are (x_0, y_0), α, a, and k. These parameters are described at length in Sec. II.A. With the exception of a, variations in these parameters reflect changes in observational conditions. Image synthesis begins with an empty array of pixel intensities $\boldsymbol{\beta}$. Where it is necessary to relate pixel indices to Cartesian coordinates, we make use of the F detector frame superposed on the array. The origin lies in the upper-left of the image. For unity-sized pixels, the coordinates of the pixel centers $\mathbf{p}$ are then

$$\mathbf{p}_{m,n_F} = \begin{bmatrix} m \\ n\end{bmatrix}$$

Each pixel corresponds to a square region. In this study, we have ignored the effects of finite fill factor, that is, spaces between pixels, but it would not be difficult to alter the simulation to include this effect.

As discussed previously, the illuminated moon comprises a rotated half-circle and half-ellipse. Because we assume that the pixels are overexposed, we model the ideal response of each pixel as

$$\boldsymbol{\beta}_{\text{ideal}} = I_{\max}\cdot E\cdot\boldsymbol{\beta}_0 \tag{21}$$

where $I_{\max}$ is the maximum response of the detector analog to digital converter (ADC), E is the degree of overexposure , and $\boldsymbol{\beta}_0$ is the normalized illumination of each pixel. The quantity E is most easily understood by considering that moonlight illuminating $1/E$ of a pixel's area will just saturate the detector response. The last quantity, $\boldsymbol{\beta}_0$, is the fraction of any particular pixel that lies within the specified half-circle/half-ellipse composite shape. Some care must be taken in its calculation. Exact calculation of the intersection areas is important to establish a realistic moon image near the edge of the moon. Because the pixels are overexposed, if the moon covers only a small fraction of a pixel, the pixel response may still be significant.

The normalized pixel illumination can be separated into components from the semicircle $\boldsymbol{\beta}_{sc}$ and semi-ellipse $\boldsymbol{\beta}_{se}$. We write

$$\boldsymbol{\beta}_0 = \begin{cases}\boldsymbol{\beta}_{sc} - \boldsymbol{\beta}_{se} & 0\le k\le 0.5 \\ \boldsymbol{\beta}_{sc} + \boldsymbol{\beta}_{se} & 0.5 < k \le 1\end{cases} \tag{22}$$

These components can be decomposed further:

$$\boldsymbol{\beta}_{sc} = \boldsymbol{\beta}_c\cdot\boldsymbol{\beta}_L \tag{23}$$

$$\boldsymbol{\beta}_{se} = \boldsymbol{\beta}_e\cdot\boldsymbol{\beta}_{L'} \tag{24}$$

where $\boldsymbol{\beta}_c$ and $\boldsymbol{\beta}_e$ are the fractions of the pixel areas that intersect the circular and elliptical regions of the moon, respectively. The $\boldsymbol{\beta}_L$ and $\boldsymbol{\beta}_{L'}$ quantities reflect the fraction of the pixel area in the appropriate half of the half-plane defined by the line of cusps (i.e., the major axis of the ellipse). This reduces the illumination calculations to the following three geometry problems: 1) the area of intersection between a rectangle (i.e., a pixel) and a circle, 2) the area of intersection between a rectangle and a rotated ellipse, and 3) the area of intersection between a rectangle and a half-plane defined by a linear boundary. None of these solutions are particularly complex, however, most involve numerous intersection cases that must be handled appropriately. In the interest of brevity, we provide only a sketch of our approach.

1. *Rectangle–Circle Intersection*

Our solution to the rectangle–circle intersection problem is based on a set of image processing libraries by Buie[‡]; an alternate approach can be found in [10]. To calculate the exact intersection between a circle of radius a and a rectangular region centered on $\mathbf{p}_{m,n}$, we first translate the coordinate system so that the origin lies at the center of the circle:

$$\mathbf{p}_G = \begin{bmatrix} p_x \\ p_y \end{bmatrix}_G = \mathbf{p}_F - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}_F \tag{25}$$

We then construct four triangles connecting the vertices of the rectangle and the origin (Fig. 9). The four vertices of each pixel $\mathbf{x}_v$ are displaced from $\mathbf{p}_F$ by half the pixel dimension δ[§]:

$$\mathbf{x}_\mathbf{v} = \begin{bmatrix} A_x & B_x & C_x & D_x \\ A_y & B_y & C_y & D_y \end{bmatrix} = \begin{bmatrix} p_x - \frac{\delta}{2} & p_x - \frac{\delta}{2} & p_x + \frac{\delta}{2} & p_x + \frac{\delta}{2} \\ p_y - \frac{\delta}{2} & p_y + \frac{\delta}{2} & p_y + \frac{\delta}{2} & p_y - \frac{\delta}{2} \end{bmatrix} \tag{26}$$

In the interest of notational simplicity, we label the individual vertices counterclockwise: A, B, C, and D. We define the notation I_{PQ} as the *signed* intersection area between the circle and $\triangle POQ$. The area is positive if the traversal from P to Q is counterclockwise, negative if clockwise, and zero if directly toward or away from O. We can the find the net intersection area of a pixel with a counterclockwise circuit of its vertices, that is,

$$\boldsymbol{\beta}_c = I_{AB} + I_{BC} + I_{CD} + I_{DA} \tag{27}$$

Thus, the circle–rectangle intersection can be reduced to several simpler intersections. These circle–triangle intersections span a number of distinct cases, but all of the solutions can be expressed in terms of triangles, chords, and sectors.

2. *Rectangle–Ellipse Intersection*

The rectangle–ellipse intersection problem is more complex than the circular case. The region of interest is no longer rotationally symmetric, and the major axis of the ellipse is angled with respect to the principal axes of the coordinate system. However, through a series of geometric transformations, we can reuse the circle–triangle solution discussed previously.

Our approach is depicted graphically in Fig. 10. Starting with the vertices $\mathbf{x}_\mathbf{v}$, we apply a rotation through an angle α, to bring the vertices into the moon-limb detector frame. This frame is aligned with the principal axes of the ellipse. Thus,

$$\mathbf{x}'_\mathbf{v} \equiv \mathbf{x}_{\mathbf{v}L} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \mathbf{x}_{\mathbf{v}G}$$

[‡]Data available at http://www.boulder.swri.edu/~buie/idl/index.html.

[§]Using δ, the calculations are still correct when the fill factor is less than unity.

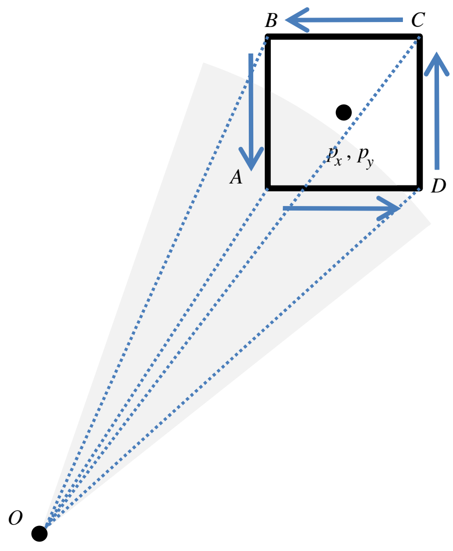

Fig. 9 Circle–rectangle intersection.

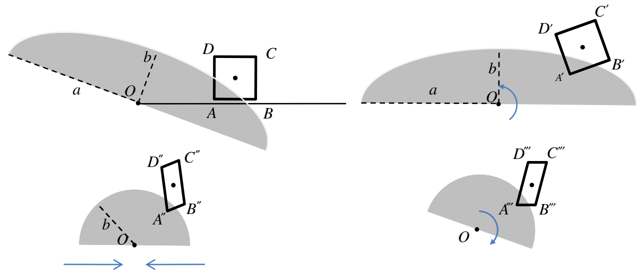

Fig. 10 Ellipse–rectangle transformations.

We then apply an affine transformation to scale the x dimension of this frame by a factor of $|b|/a$. This transforms a square pixel into a parallelogram and the ellipse into a circle of radius $|b|$. The triangle–circle intersection algorithms described previously are optimized for horizontal or vertical edges. To use these efficient routines, we rotate the edges of the parallelogram to bring them parallel to the x or y axes. Because the ellipse has been scaled into a circle, this rotation does not affect the area of intersection. The two pairs of edges must be rotated separately. Using plane geometry, we can show

$$\tan\alpha' = \begin{cases} -\frac{|b|}{a}\tan\alpha & \text{for } AB, CD \\ -\frac{a}{|b|}\tan\alpha & \text{for } BC, AD \end{cases} \tag{28}$$

We note that both α and α' must lie in the same quadrant. This series of transformations coverts the ellipse–rectangle intersection into a circle–rectangle interaction. We calculate the intersection in a manner similar to Eq. (27), but we must multiply the result by $a/|b|$ to compensate for the area distortion introduced by the affine scaling (the intersection area is trivially zero, when $b = 0$):

$$\boldsymbol{\beta}_e|_{m,n} = \frac{a}{|b|}(I_{A'''B'''} + I_{B'''C'''} + I_{C'''D'''} + I_{D'''A'''})|_{m,n} \tag{29}$$

3. *Half-Plane–Rectangle Intersection*

The semi-ellipse and semicircle regions are defined by a combination of conic and linear boundaries. The linear boundary is defined by the line of cusps. The equation for this line can be given in the moon-centroid frame of reference G:

$$y_G = \tan(\alpha)x_G \tag{30}$$

Formally, the region of interest becomes an inequality whose sense depends on α. For the circular case, we have

$$y_G \leq \tan(\alpha)x_G \qquad -\frac{\pi}{2} \leq \alpha < \frac{\pi}{2}$$
$$y_G \geq \tan(\alpha)x_G \qquad \alpha < -\frac{\pi}{2}, \qquad \alpha > \frac{\pi}{2} \tag{31}$$

The semi-ellipse is similar: the half-plane regions defined in Eq. (31) remain unchanged for $0 \leq k \leq 0.5$, but for $0.5 < k \leq 1$, we reverse the two α cases.

Once we have determined the appropriate boundary, we must calculate the fraction of each pixel in the region of interest. The intersections for the semicircular portion of the moon disk are denoted $\boldsymbol{\beta}_L$; the semi-ellipse regions are denoted $\boldsymbol{\beta}_{L'}$. A number of intersection cases must be considered between the rectangular pixels and the boundary line. Depending on how the boundary cuts through a given pixel, we will need to calculate the area of a rectangular, triangular, or trapezoidal region.

4. *Ideal Image Synthesis*

When the circle, ellipse, and half-plane components of the moon shape are combined using the elementwise operations of Eqs. (22–24), we are left with a continuous map of the illuminated portion of each pixel:

$$\boldsymbol{\beta}_0 = \begin{cases} \boldsymbol{\beta}_c\boldsymbol{\beta}_L - \boldsymbol{\beta}_e\boldsymbol{\beta}_{L'} & 0 \leq k \leq 0.5 \\ \boldsymbol{\beta}_c\boldsymbol{\beta}_L + \boldsymbol{\beta}_e\boldsymbol{\beta}_{L'} & 0.5 < k \leq 1 \end{cases} \tag{32}$$

For a gibbous moon image, interior pixels along the major axis will be counted fully because the linear boundaries used to generate $\boldsymbol{\beta}_L$ and $\boldsymbol{\beta}_{L'}$ yield complementary intersection areas. For crescent moon images, the major axis pixels will generally report zero contributions, because the identical linear boundary contributions are subtracted.

B. Focus Blur

The pixel fill calculations from Eq. (32), represent ideal pixel illumination from a perfectly delimited moon disk. The raw synthesized image is not truly realistic because physical limitations bound achievable resolution in optical devices. The point-spread characteristics of the optics redistribute incident energy over areas of the detector that might otherwise be unilluminated. The lower bound on this blur is determined by diffraction, but it is a common practice to intentionally defocus star-tracker optics to improve star centroid calculation.

We approximate the effects of this phenomena by convolving a two-dimensional truncated Gaussian filter with the raw image determined in Eq. (32). This filter has impulse response $\mathbf{h}_{G_{m,n}}$, and is characterized by the peak width σ_G and the size of the filter N_G. Thus,

$$\boldsymbol{\beta}_{\text{blur}} = \boldsymbol{\beta}_{\text{ideal}} \otimes \mathbf{h}_G \tag{33}$$

For the tests presented in this paper, we have used $\sigma_G = 0.5$ and $N_G = 5$. This represents a fairly small amount of blur. In practice, we recommend choosing σ_G based on the target sensors' optical characteristics; a suitable value of N_G can then be chosen based on the image exposure and expected threshold levels.

C. Pixel Noise, Saturation, and Quantization

Several distinct physical phenomena contribute to the noise observed in pixel detectors (see [11] for a discussion). Each has its own spatial and temporal statistical characteristics. To keep our image synthesis tractable, we consider only a model of dark current noise. The noise response $\boldsymbol{\beta}_{N_{m,n}}$ for each pixel is a combination of a constant, scalar offset I_{DC_0}, and a normally distributed fluctuation $\boldsymbol{\beta}_{DC_{m,n}}$. This latter component is described by its standard deviation σ_{DC} and represents both spatial (e.g., pixel-to-pixel), and temporal (i.e., image-to-image) variations. Thus,

$$\boldsymbol{\beta}_{N_{m,n}} = I_{DC_0} + \boldsymbol{\beta}_{DC_{m,n}} \tag{34}$$

If we assume that these noise parameters are normalized with respect to the maximum detector response, we can combine the effects of image synthesis, blur, and noise given in Eqs. (21), (33), and (34):

$$\boldsymbol{\beta}_{m,n} = I_{\max} \cdot (E \cdot \boldsymbol{\beta}_{0_{m,n}} \otimes \mathbf{h}_{G_{m,n}} + I_{DC_0} + \boldsymbol{\beta}_{DC_{m,n}}) \tag{35}$$

Based on our experience with our baseline camera, $I_{DC_0} = 0.052$ and $\sigma_{DC} = 9 \times 10^{-3}$ are reasonable values for the noise parameters.

The final step in the image synthesis process is to capture the effect of finite ADC resolution and range. If $I_{\max}$ is expressed in detector counts, then these effects can be implemented by truncating the decimal part of $\boldsymbol{\beta}$ and enforcing the saturation limit:

$$\boldsymbol{\beta}'_{m,n} = \begin{cases} \lfloor \boldsymbol{\beta}_{m,n} \rfloor & \boldsymbol{\beta}_{m,n} < I_{\max} \\ I_{\max} & \boldsymbol{\beta}_{m,n} \geq I_{\max} \end{cases} \tag{36}$$

Our approach to noise injection is, admittedly, crude. It neglects effects such as shot, fixed-pattern, readout, and reset noise: all of which will affect the true image formation. Some justification is needed for neglecting these effects. First, because the synthetic images are overexposed, any saturated pixels will still be saturated after applying noise. Thus, interior pixels in the moon disk are unchanged. Furthermore, the noise experienced by dark pixels, namely the background around the illuminated moon, is well characterized by our dark current model. Therefore, we contend that the response of these pixels is also well approximated. Hence, we expect the largest discrepancies between synthetic and real images to occur in the partially illuminated pixels on the moon's perimeter where shot noise may become significant. However, our processing algorithms use pixel intensity only for thresholding purposes. Only those pixels that may have been pushed over this processing threshold by unmodeled noise will appear incorrectly in subsequent processing. Thus, our noise modeling is not unreasonable.

IV. Processing Algorithms

Attitude measurement using the techniques outlined in Sec. II.D relies on accurate identification of the moon position and orientation within an image. Our approach to this problem proceeds in stages. We first present algorithms to find initial estimates of the moon parameters (e.g., $\tilde{\alpha}$, $\tilde{k}$, etc.; the tilde denotes an estimate), and then refine these values using a nonlinear least-squares fit. The processing begins with a synthetic moon image generated according to the models developed in Sec. III. This development assumes three simple preliminary processing steps that will not be addressed in this paper. First, that a routine exists that is able to identify a moon-in-view condition and initiate the moon-tracking mode. Second, that the exposure time has been adjusted to yield the desired level of overexposure E. Third, that the moon position can be roughly constrained to a window of about 1.3 times the expected diameter of the moon. This final assumption permits us to work with smaller image structures but also carries the implicit assumption that the moon is the only illuminating body in the windowed FOV.

Our aim in this section is to present a comprehensive series of algorithms useful for extracting and refining *all* of the moon shape parameters directly from star-tracker images. Because some of these parameters may be determined a priori from ephemerides and other sources, we also discuss some of the variations to this processing framework that could be considered during implementation. Some discussion of the tradeoffs between estimation accuracy and computational cost are presented in Sec. V.

A. Image Preprocessing

Before extracting the moon parameters from the moon image, we must first perform several processing steps on the moon image array $\boldsymbol{\beta}'$. This study employs images synthesized using Eq. (36), but a deployed system would use actual camera images. We must threshold the intensity values, identify tight constraints on the moon position, estimate the first image parameter k, and locate perimeter points in the image. We first choose a threshold intensity I_{thresh} and define the thresholded image $\boldsymbol{\beta}_T$:

$$\boldsymbol{\beta}_{T_{m,n}} = \begin{cases} \boldsymbol{\beta}'_{m,n} & \boldsymbol{\beta}'_{m,n} \geq I_{\text{thresh}} \\ 0 & \text{otherwise} \end{cases} \tag{37}$$

Because we have overexposed the image, we can set the threshold quite high. The baseline camera used in this study has 12-bit ADC in the detector (i.e., $I_{\max} = 4095$) and most of the simulations use $I_{\text{thresh}} = 4000$. A binary version of the thresholded image is also useful:

$$\boldsymbol{\beta}_{B_{m,n}} = \begin{cases} 1 & \boldsymbol{\beta}'_{m,n} \geq I_{\text{thresh}} \\ 0 & \text{otherwise} \end{cases} \tag{38}$$

Our next preprocessing step is to establish tight bounds on the moon within the image. We are looking for the row and column bounds that frame all the pixels with intensity greater than I_{thresh}. Formally, the row bounds $m_{\min}$ and $m_{\max}$ are defined by

$$m_{\min} = \min_{m}(\exists n | \boldsymbol{\beta}_{B_{m,n}} \geq 0) \tag{39}$$

and

$$m_{\max} = \max_{m}(\exists n | \boldsymbol{\beta}_{B_{m,n}} \geq 0) \tag{40}$$

Similar expressions define the column bounds $n_{\min}$ and $n_{\max}$.

The illuminated moon fraction k can be interpreted as an area ratio, and so we can estimate this parameter from the total intensity in the windowed image:

$$\tilde{k} = \frac{\sum_{m=m_{\min}}^{m_{\max}} \sum_{n=n_{\min}}^{n_{\max}} \boldsymbol{\beta}_{T_{m,n}}}{I_{\max} \cdot \pi a^2} \tag{41}$$

This calculation relies on knowing a, a quantity we have not yet estimated. Because $\tilde{k}$ serves as a guess for future refinement, it is usually possible to use the maximum value of a that we expect to see over the course of a mission. Such an assumption will introduce a bias in the calculation of $\tilde{k}$. Additionally, overexposure creates excess illumination in the image when the moon is nearly full. Thus, it is also important that we force $\tilde{k} \leq 1$. Our simulation results show no evidence that these restrictions affect the overall algorithm accuracy.

Although the illuminated moon fraction k can be calculated a priori from lunar ephemeris (Sec. II.D), it is simple to obtain a rough estimate from the image itself. The image-derived value of $\tilde{k}$ could be used during anomalous operational conditions (e.g., safe-mode recovery) when onboard ephemerides may be untrustworthy.

Overexposing the moon image removes surface details from the face of the moon. This leaves the boundary between the bright moon and the dark background as the only feature that can be used to locate and orient the moon. Before we can continue the image processing, we must identify the points $\mathbf{x}_s$ that comprise the perimeter of the moon in the image. Except where otherwise noted, the N_s points in $\mathbf{x}_s$ are implicitly expressed in F. The pixilated image is an approximation of the underlying smooth shape, and so the perimeter is also an approximation. We consider two definitions of perimeter points: one based on pixel centers and one based on pixel vertices.

Our first heuristic for finding the moon perimeter starts by taking pixel centers as the perimeter points. If m_i, n_i are the indices of the ith perimeter pixel, then the perimeter point has coordinates

$$\mathbf{x}_{si} = \mathbf{p}_i = \begin{bmatrix} m_i \\ n_i \end{bmatrix} \tag{42}$$

To find the pixels on the edge of the moon shape, we must identify the foreground pixels (i.e., those where $\boldsymbol{\beta}_{B_{m,n}} = 1$) that are *connected* to background pixels (i.e., those where $\boldsymbol{\beta}_{B_{m,n}} = 0$). Two types of adjacency typically considered in image processing are four-connected and eight-connected pixels (Fig. 11). The former checks only in the four principal directions (i.e., up, down, left, right), whereas the latter checks the diagonal pixels as well.

Depending on the values of E and I_{thresh}, this strategy may consistently bias the perimeter estimates toward points that are actually inside or outside of the continuous shape. For instance, if

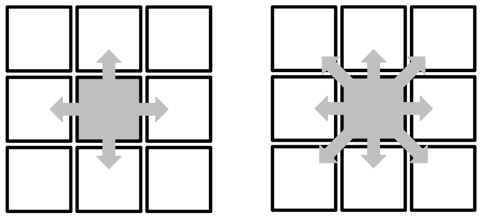

Fig. 11 Pixel connectivity.

$E = 1$ and $I_{\text{thresh}} = I_{\max}$, then all foreground pixels will necessarily lie entirely within the continuous region, and the perimeter $\mathbf{x}_s$ coordinates will likewise lie inside of the true perimeter. Conversely, high values of E (or low values of I_{thresh}) will tend to push the $\mathbf{x}_s$ values outside of the perimeter.

Using the pixel *vertices* rather than pixel centers can yield a set of points more evenly distributed on both sides of the continuous moon boundary. Once we have identified the perimeter pixels, we then select a subset of the vertices of $\mathbf{p}_i$. The vertices are found from Eq. (26), so that instead of Eq. (42), the perimeter points become

$$\mathbf{x}_s \subseteq \bigcup_i \mathbf{x}_{vi} \tag{43}$$

For each foreground pixel, we select those vertices that are connected to neighboring background pixels. Figure 12a illustrates the connection topology. Because some vertices are shared between neighboring pixels, care must be taken in generating $\mathbf{x}_s$ to avoid including duplicate points. This can be accomplished in a single algorithmic pass through the image array by recognizing that the iteration direction establishes a dominance relation between pixels (Fig. 12b). A vertex $\mathbf{v}$ of foreground pixel i will have already been added to $\mathbf{x}_s$ if any of the dominating pixels that share $\mathbf{v}$ are also foreground pixels.

In summary, we may define our perimeter pixels $\mathbf{x}_s$ using either Eq. (42) or Eq. (43), namely, the centers or vertices of the foreground pixels in $\boldsymbol{\beta}_B$. Using pixel vertices generally yields points closer and better distributed about the true moon boundary. Conversely, the pixel center approach is computationally simpler and, in the four-connected case, yields substantially fewer perimeter points than the alternatives. In our tests, this amounts to about a 30–40% fewer perimeter points. Reducing the number of perimeter points will reduce the computational requirements during later processing, but it is unclear how the choice of strategy will affect the moon-tracking accuracy.

B. Moon Center and Orientation

Once we have extracted an acceptable set of perimeter pixels $\mathbf{x}_s$, we must estimate the remaining image parameters, namely, x_0, y_0, and α. Identifying the semicircular limb edge of the moon is a crucial part of generating an initial estimate of the moon's position. Park and Mortari [2] first calculate $\tilde{\alpha}$ from the second moments of the image, and then find the moon center using a two-dimensional cross correlation between the captured, partially illuminated moon image and a rotated reference image of the full-moon disk. From our initial investigations, this latter step has limited utility for our system.

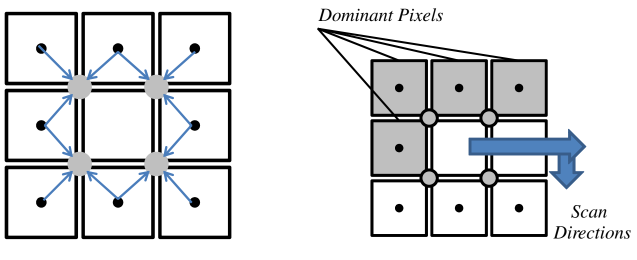

Fig. 12 Vertex connectivity.

Overexposing the captured images simplifies the required processing in some respects but eliminates the image features on the moon's face that would permit correlation-based registration. Cross correlations between the captured, partially illuminated images and reference circular disks show substantial ambiguity. Displacement in the direction of the line of cusps is well resolved, but perpendicular motions (i.e., toward or away from the limb), yield a very broad correlation peak. Instead, we adopt a different approach to finding initial estimates for the moon center and orientation: the minimum spanning circle (MSC).

1. *Minimum Spanning Circle*

Finding the minimum radius circle that encloses a given set of points is a classical problem in computational geometry and operations research. Original solutions proposed by Sylvester [12] over 150 years ago still form the basis of many modern algorithms [13]. The parallels between the classical MSC problem and our moon-tracking application are clear; perimeter points from the limb yield the circular solution, whereas those on the terminator will lie completely within the minimum radius circle, that is, the moon disk. The algorithm calculates the moon center estimate $(\tilde{x}_0, \tilde{y}_0)$, but also provides an estimate of the moon radius $\tilde{a}$.

In this study, we have adapted the MSC algorithm by Oommen [14]. For the sake of brevity, we will not reproduce the full algorithm here, but instead provide some comments on performance. This algorithm considers triplets of points from $\mathbf{x_s}$. Starting with a large circle guaranteed to enclose all the points, the algorithm iteratively shrinks the candidate circle, adjusting the origin estimate as it proceeds. The algorithm is guaranteed to converge in $O(\log N_s)$ iterations, each of $O(N_s)$ complexity. Efficient culling of the perimeter points keeps the coefficients to these terms quite low; in our trials, with $N_s \approx 150$, the MSC algorithm converged consistently in 5–6 iterations. The estimates of x_0, y_0, and a derived from this method were quite good.

To some extent, this algorithm appears unnecessarily complex; because the star-tracker optics effectively define a, there is little need to estimate this quantity. In practice, the ambiguity in establishing the true perimeter of the moon image, combined with focus blur may make a priori calculation of a inaccurate. Allowing a to vary makes our approach robust to a variety of nonideal image characteristics.

2. *Estimating Moon Rotation*

The last quantity that must be estimated is the rotation angle α. To obtain an estimate of this quantity, we calculate the brightness centroid $(x_c, y_c)_F$ of the thresholded moon image $\boldsymbol{\beta}_T$:

$$x_{c_F} = \frac{\sum_m \sum_n n \cdot \boldsymbol{\beta}_{T_{m,n}}}{\sum_m \sum_n \boldsymbol{\beta}_{T_{m,n}}} \quad \text{and} \quad y_{c_F} = \frac{\sum_m \sum_n m \cdot \boldsymbol{\beta}_{T_{m,n}}}{\sum_m \sum_n \boldsymbol{\beta}_{T_{m,n}}} \tag{44}$$

Recall the geometry of the moon image (Fig. 3). From symmetry, the brightness centroid should lie on the $\hat{\mathbf{L}}_y$ axis, and α can be estimated by considering the moon center and the centroid location. If r_α is the distance between these two points,

$$r_{\alpha_F} = \sqrt{(x_{c_F} - \tilde{x}_{0_F})^2 + (y_{c_F} - \tilde{y}_{0_F})^2} \tag{45}$$

then it is easy to show that

$$\cos\alpha = \frac{y_{c_F} - \tilde{y}_{0_F}}{r_{\alpha_F}} \quad \text{and} \quad \sin\alpha = \frac{x_{0_F} - \tilde{x}_{c_F}}{r_{\alpha_F}} \tag{46}$$

A four-quadrant arctangent function will calculate α directly from the centroid and center points directly.

C. Nonlinear Least-Squares Fitting

The heuristics presented in Sec. IV.B provide initial estimates of the moon's orientation, position, and illumination in the star-tracker image. Although this represents sufficient information to calculate the sensor orientation, the resulting attitude accuracy may not be sufficient. To improve our estimates, we wish to minimize the mean-squared distance between the points in $\mathbf{x_s}$ and the composite semicircle/semi-ellipse shape of the moon's perimeter. We perform this minimization using a standard iterative Newton–Euler nonlinear least-squares formulation. Given a vector of parameters

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & \cdots & u_{M_f} \end{bmatrix}^T \tag{47}$$

we seek to minimize a cost function of the form

$$\Psi = \sum_{i=1}^{M} f_i(\mathbf{u})^2 \tag{48}$$

In typical curve-fitting applications, each perimeter point contributes to the squared error, $M = M_s$, and the f_is represent the Euclidean distance between the observed perimeter points and the modeled curve. The cost function components f_i can also be written in vector form:

$$\mathbf{f}(\mathbf{u}) \equiv \begin{bmatrix} f_1(\mathbf{u}) \\ f_2(\mathbf{u}) \\ \vdots \\ f_M(\mathbf{u}) \end{bmatrix}$$

If $J(\mathbf{u})$ is the Jacobian of f, namely,

$$\mathbf{J}(\mathbf{u}) = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \cdots \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \\ \vdots & & \ddots \end{bmatrix} \tag{49}$$

then, at each iteration j, we must solve the *linear* least-squares problem:

$$\mathbf{J}(\mathbf{u_j})\mathbf{du_j} = -\mathbf{f}(\mathbf{u_j}) \tag{50}$$

to find the correction vector $\mathbf{du}$ which we apply to $\mathbf{u}$:

$$\mathbf{u}_{\mathbf{j}+1} = \mathbf{u_j} + \mathbf{du_j} \tag{51}$$

To apply this approach to the moon-tracking problem, we must structure our analytic model of the moon shape within this framework. We have developed a intuitive and simple parametrization to describe moon shape and position, namely, a, k, α, x_0, and y_0. Exploiting this existing analysis is a natural choice for the least-squares model. Numerous researchers have studied the least-squares fitting of circles and ellipses; we base our development primarily on the work of Gander et al. [15]. Here, we sketch out their model formulation and extend their method to accommodate the composite shape of the moon image.

We first consider fitting an ellipse to a set of Cartesian points $\mathbf{x_e}$. An implicit formula for an elliptical curve can be written in terms of the same five parameters we have selected in the last paragraph, although the semiminor axis b is more commonly used instead of k. Calculating the distance from a point to a circle is a trivial operation, however, calculating the distance to an ellipse requires solving a quartic equation. To avoid the explicit solution of these quartic equations for each of the M_e points in $\mathbf{x_e}$, we employ a parametric equation for the ellipse:

$$\mathbf{x}_F = \mathbf{x}_{0F} + \mathbf{C}_{GL}(\alpha)\mathbf{x}'_L \tag{52}$$

where

$$\mathbf{x}'_L = \begin{bmatrix} a\cos\varphi \\ b\sin\varphi \end{bmatrix} \quad \text{and} \quad \mathbf{C}_{GL}(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \tag{53}$$

We recognize the $\mathbf{x_0}$ offset is the translation between G and F. Continuing our derivation, we temporarily drop the reference frame subscript. To make use of this parametric equation, we must augment the model parameters with a set of φ_i, each corresponding to one of the points in $\mathbf{x_e}$. The least-squares cost function is constructed from components of the form

$$\mathbf{g}_i = \begin{bmatrix} x_{e_i} \\ y_{e_i} \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} - \mathbf{C}_{GL}(\alpha) \begin{bmatrix} a\cos\varphi_i \\ b\sin\varphi_i \end{bmatrix} \tag{54}$$

Thus, the M_e component $\mathbf{g}_i$s are concatenated in a column vector to form the error vector $\mathbf{f}$:

$$\mathbf{f} \equiv \mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \end{bmatrix}$$

We can see that, when the φ_i are chosen appropriately, $\|\mathbf{f}\|^2$ is equivalent to the sum of squares of the distances between the ellipse and $\mathbf{x_e}$. The parameter vector is a concatenation of the φ_i and the standard parameters:

$$\mathbf{u} = \begin{bmatrix} \varphi_1 & \varphi_2 & \dots & \varphi_{M_e} & \tilde{\alpha} & \tilde{a} & \tilde{b} & \tilde{x}_0 & \tilde{y}_0 \end{bmatrix}^T \tag{55}$$

If we define the $2M_f \times 2M_f$ block-diagonal matrix $\mathbf{U}$

$$\mathbf{U} = -\mathrm{diag}(\mathbf{C}_{GL}) \tag{56}$$

Writing Eq. (50) in terms of $\mathbf{g}$ and premultiplying by $\mathbf{U}^T$ gives

$$\mathbf{U}^T\mathbf{J}(\mathbf{u})\mathbf{du} = \mathbf{U}^T\mathbf{g} \tag{57}$$

or, more simply,

$$\mathbf{J}'(\mathbf{u})\mathbf{du} = \mathbf{g}' \tag{58}$$

If we evaluate the partial derivatives in the Jacobian, and permute the rows to separate the x- and y-axis contributions, we can write J' in block form:

$$\mathbf{J}' = \begin{bmatrix} -a\mathbf{S} & \mathbf{A} \\ b\mathbf{D} & \mathbf{B} \end{bmatrix} \tag{59}$$

where

$$\mathbf{S} = \mathrm{diag}(\sin\varphi_i), \qquad \mathbf{D} = \mathrm{diag}(\cos\varphi_i) \tag{60}$$

These are both square $M_e \times M_e$ matrices. The rows of the $M_e \times 5$ element $\mathbf{A}$ and $\mathbf{B}$ matrices are given by

$$\begin{aligned} \mathbf{A}_i &= [-b\sin\varphi_i \quad \cos\varphi_i \quad 0 \quad \cos\alpha \quad \sin\alpha] \\ \mathbf{B}_i &= [\ \ a\cos\varphi_i \quad 0 \quad \sin\varphi_i \quad -\sin\alpha \quad \cos\alpha] \end{aligned} \tag{61}$$

The right-hand side of Eq. (57) gives the elements of $\mathbf{g}'$:

$$\mathbf{g}'_i = \mathbf{C}^T_{GL} \begin{bmatrix} x_{e_i} - x_0 \\ y_{e_i} - y_0 \end{bmatrix} - \begin{bmatrix} a\cos\varphi_i \\ b\sin\varphi_i \end{bmatrix} \tag{62}$$

Before using these results, we must permute the entries as we did with Eq. (59) to separate the x and y components. A graphical depiction of the fit geometry is shown in Fig. 13. The deviation between the sample point and the elliptical curve are exaggerated for clarity.

Using Eq. (3) and replacing $\tilde{b}$ with $\tilde{k}$ in the parameter vector gives a slightly different model:

$$\mathbf{u_k} = \begin{bmatrix} \varphi_1 & \varphi_2 & \dots & \varphi_{m_e} & \tilde{\alpha} & \tilde{a} & \tilde{k} & \tilde{x}_0 & \tilde{y}_0 \end{bmatrix}^T \tag{63}$$

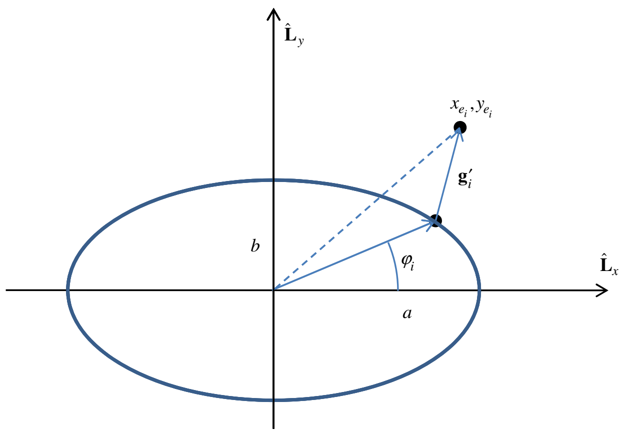

Fig. 13 Geometry for least-squares fit to ellipse.

$$\mathbf{g}'_{\mathbf{k}i} = \mathbf{C}^T_{GL} \begin{bmatrix} x_{e_i} - x_0 \\ y_{e_i} - y_0 \end{bmatrix} - \begin{bmatrix} a\cos\varphi_i \\ a(1-2k)\sin\varphi_i \end{bmatrix} \tag{65}$$

We note that, as k ranges from zero to one, b will take on values from $+a$ to $-a$. Although it is somewhat unusual to use a negative value for the semiminor axis, this allows us to restrict φ_i to $0 \le \varphi_i < \pi$; gibbous and crescent shapes are automatically resolved by the choice of k.

If we keep the same parameter formulation, we can also fit a circle to a second set of points $\mathbf{x_c}$:

$$\mathbf{J}'_{\mathrm{circ}} = \begin{bmatrix} -a\mathbf{S} & \mathbf{A} \\ a\mathbf{D} & \mathbf{B} \end{bmatrix} \tag{66}$$

$$\begin{aligned} \mathbf{A}_{\mathrm{circ}_i} &= [-a\sin\varphi_i \quad \cos\varphi_i \quad 0 \quad \cos\alpha \quad \sin\alpha] \\ \mathbf{B}_{\mathrm{circ}_i} &= [\ \ a\cos\varphi_i \quad \sin\varphi_i \quad 0 \quad -\sin\alpha \quad \cos\alpha] \end{aligned} \tag{67}$$

$$\mathbf{g}'_{\mathrm{circ}_i} = \mathbf{C}^T_{GL} \begin{bmatrix} x_{c_i} - x_0 \\ y_{c_i} - y_0 \end{bmatrix} - \begin{bmatrix} a\cos\varphi_i \\ a\sin\varphi_i \end{bmatrix} \tag{68}$$

1. *Curve Selection*

Taken together, Eqs. (59) and (64–68) form a composite model to describe the shape of the moon perimeter. For each nonlinear least-squares iteration, we synthesize $\mathbf{J}'$ and $\mathbf{g}'$, taking rows from each model. To do this, we recognize that $\mathbf{x_e}$ and $\mathbf{x_c}$ form a partition of $\mathbf{x_s}$, that is,

$$\mathbf{x_e} \cup \mathbf{x_c} = \mathbf{x_s} \tag{69}$$

and

$$\mathbf{x_e} \cup \mathbf{x_c} = \emptyset \tag{70}$$

If point $\mathbf{x}_{si}$ lies on the semi-ellipse, we use Eqs. (59), (64), and (65); if it lies on the semicircle, we use Eqs. (66–68). We do not know a priori which elements of $\mathbf{x_s}$ belong to the semi-ellipse or the semicircle, and so we evaluate $\|\mathbf{g}_{\mathbf{k}_i}\|$ and $\|\mathbf{g}_{\mathbf{circ}_i}\|$ for each point. Each $\|\mathbf{g}_i\|$ value represents the distance from the perimeter point to the modeled curve. If $\|\mathbf{g}_{\mathbf{k}_i}\| < \|\mathbf{g}_{\mathbf{circ}_i}\|$, then we conclude that $\mathbf{x}_{si}$ is a member of the semi-ellipse, otherwise $\mathbf{x}_{si}$ belongs to the points on the semicircle. This assignment must be reevaluated during each

$$\begin{aligned} \mathbf{A}_{k_i} &= [-a(1-2k)\sin\varphi_i \quad \cos\varphi_i \quad 0 \quad \cos\alpha \quad \sin\alpha] \\ \mathbf{B}_{k_i} &= [\quad a\cos\varphi_i \quad (1-2k)\sin\varphi_i \quad -2a\sin\varphi_i \quad -\sin\alpha \quad \cos\alpha] \end{aligned} \tag{64}$$

iteration because parameter changes may alter the distance between the perimeter points and the modeled curves.

2. Implementation Notes

The nonlinear curve fit requires initial estimates for each of the parameters in $\mathbf{u}$. Initial values for $\tilde{x}_0$, $\tilde{y}_0$, and $\tilde{\alpha}$ were obtained in Sec. IV.B. The sensor optical parameters, or the MSC result, determines $\tilde{a}$. Lastly, the initial $\tilde{k}$ is determined from either ephemeris predictions or Eq. (41). Initial estimates for the remaining parameters φ_i can be obtained from the rotated coordinates of the perimeter points, that is,

$$\begin{bmatrix} x_{i_L} \\ y_{i_L} \end{bmatrix} = \mathbf{C}_{GL}^T \begin{bmatrix} x_{i_F} - x_{0_F} \\ y_{i_F} - y_{0_F} \end{bmatrix} \tag{71}$$

Provided this transformation brings the points close to the modeled curves, then a reasonable initial estimate for the φ_i will be (from Fig. 13)

$$\tilde{\varphi}_i = \arccos\left(\frac{x_{i_L}}{\sqrt{x_{i_L}^2 + y_{i_L}^2}}\right) \tag{72}$$

The arccosine is sufficient to determine the angles φ_i because we have enforced that $0 \le \varphi_i < \pi$.

The *linear* least-squares system of Eq. (58), can be solved with any appropriate numerical algorithm. A sparse matrix representation of the Jacobian is appropriate due to the large number of zero elements. In this study, the standard Matlab sparse solver was used (i.e., QR decomposition with pivoting), but the choice of algorithm can be revisited for embedded implementations. The parameter updates are applied to the $\mathbf{u_k}$ vector and the next iteration begins. The algorithm terminates once the system has converged or a maximum number of iterations has been reached. Typically, the convergence criterion for nonlinear least-squares problems is based on the infinity norm of the update vector $\mathbf{du}$.

In our trials, the initial estimates usually resulted in good convergence behavior (i.e., $\|\mathbf{du}\|_\infty$ dropped to less than 10^{-6} within a handful of iterations), despite the large number of parameters in the fit. Several heuristics were adopted to improve the numerical robustness of the solver in the face of problematic images. First, the valid ranges of parameters such as k and φ_i were strictly enforced. This check will detect a divergent solution, but will not improve stability if the initial estimates are poor or the problem is ill conditioned. More important, during each algorithm iteration, the $\mathbf{g}_i'$ and $\mathbf{g}_{\text{circ}_i}'$ values were calculated using both the current values of φ_i, as well as a recalculated set of φ_i guesses from Eq. (72). During each iteration, we choose the closest boundary curve and best φ_i values for each perimeter point. This resets the values of φ_i if they are driven away from good values by poor conditioning. Because the parameters are only changed if the new guesses will improve the least-squared error at the current iteration, this approach cannot have a negative effect on convergence. This decision logic relies only on the current parameter estimates and is appropriate for online use. Although calculating four variations of the $\mathbf{g}$ vector in each iteration introduces some mathematical inefficiency, this part of the processing is not a major contributor to the overall computational cost of the algorithms.

V. Simulation and Results

In the preceding sections, we have presented a comprehensive and expansive treatment of the image analysis required to obtain attitude information from overexposed images of the moon. Our full solution approach starts with a set of coarse parameter estimates extracted from the moon image using straightforward image analysis and refines these quantities with a nonlinear least-squares model of the moon shape. In this section, we evaluate the merits of this processing approach using a series of simulation experiments. The first set of tests considers several image processing variations appropriate for our baseline sensor design. We examine the importance of each processing stage and evaluate the effectiveness of our initial estimates. The second set of tests briefly examines how different operating conditions and sensor designs will affect the algorithm performance.

For each experiment, we simulate a large number of test images with different values of k, x_0, y_0, and α. Our baseline optical design sets $a = 30$ pixels. Through these tests, we wish to establish how performance varies with the phase of the moon. Thus, k is varied over the range $0.02 \le k \le 0.98$. For each value of k, we generate $N_{\text{trial}} = 100$ test images. The remaining image parameters are selected randomly for each test image from uniform distributions. We have assumed that the moon can be coarsely located by an external routine, and so we do not need to generate a full resolution detector image, merely a windowed image slightly larger than the moon disk. Hence, it is sufficient to vary x_0 and y_0 within a range of only a few pixels. The moon orientation α is distributed over $0 \le \alpha < 2\pi$. The simulation is not intended to replicate a particular orbit, but merely to evaluate the range of moon shapes that a star tracker might see.

Inaccuracy in estimating the moon center in the image is directly related to the error in the satellite–moon vector $\hat{\mathbf{s}}_{TM_T}$. To convert the image-plane error into an angular error, we scale the former by the apparent size of the moon and multiply by the moon's true *angular* radius. For a single test image taken on Earth (or in low Earth orbit), the angular error is

$$\Delta_m = \frac{\sqrt{(\tilde{x}_0 - x_0)^2 + (\tilde{y}_0 - y_0)^2}}{a} \cdot \arctan\frac{r_{\text{moon}}}{R_{em}} \tag{73}$$

For simplicity, we ignore any variation in moon size caused by optical distortions or the satellite's orbit. Assuming that the x and y errors are uncorrelated and distributed normally, Δ_m will follow a Rayleigh distribution with mean σ_m:

$$\sigma_m^2 = \frac{\sum_i \Delta_{m_i}^2}{N_{\text{trial}}} \tag{74}$$

The underlying standard deviation in x and y, σ_{XY}, is closely related to σ_m:

$$\sigma_{XY} = \sqrt{\frac{\pi}{2}}\sigma_m \tag{75}$$

We can also calculate the error in $\tilde{\alpha}$:

$$\Delta_\alpha = \tilde{\alpha} - \alpha \tag{76}$$

The angular error in the satellite–sun vector depends on Δ_α, but also on the phase angle ξ. For small Δ_m (an assumption that was borne out in most of our testing), we can approximate the error in the satellite–sun vector from

$$\Delta_s \approx \sin\xi\Delta_\alpha \tag{77}$$

Consequently, we can relate the standard deviations of Δ_α and Δ_s

$$\sigma_s = \sin\xi\sigma_\alpha \tag{78}$$

Using Eq. (2), we can express this in terms of k:

$$\sigma_s = \sqrt{4k(1-k)}\sigma_\alpha \tag{79}$$

If the moon and sun vectors are colinear, that is, when $k = 1$, the satellite attitude cannot be determined from $\hat{\mathbf{s}}_{TM}$ and $\hat{\mathbf{s}}_{MS}$.

A. Effect of Processing Approach

Throughout the development of our processing strategy, we have noted that some of our heuristics are redundant: lunar ephemeris calculations can predict k and our optical and orbital design determines a. In this section, we evaluate the performance of our baseline processing approach and consider several variants that use different model parametrization.

Table 3 Solution variants

Test name	$\mathbf{u}^T$	Initial a	Initial k	Perimeter points
`Full`	$\begin{bmatrix}\varphi_1 & \varphi_2 & \dots & \varphi_{m_e} & \tilde{\alpha} & \tilde{a} & \tilde{k} & \tilde{x}_0 & \tilde{y}_0\end{bmatrix}$	Image	Image	Vertices
`InitialEst`	N/A	Image	Image	Vertices
`FullCenter`	$\begin{bmatrix}\varphi_1 & \varphi_2 & \dots & \varphi_{m_e} & \tilde{\alpha} & \tilde{a} & \tilde{k} & \tilde{x}_0 & \tilde{y}_0\end{bmatrix}$	Image	Image	Centers
`Restricted`	$[\tilde{\alpha} \quad \tilde{a} \quad \tilde{k} \quad \tilde{x}_0 \quad \tilde{y}_0]$	Image	Image	Vertices
`EphemEst`	$[\varphi_1 \quad \varphi_2 \quad \dots \quad \varphi_{m_e} \quad \tilde{\alpha} \quad a \quad k \quad \tilde{x}_0 \quad \tilde{y}_0]$	Ephemeris	Ephemeris	Vertices
`EphemConst`	$[\varphi_1 \quad \varphi_2 \quad \dots \quad \varphi_{m_e} \quad \tilde{\alpha} \quad \tilde{x}_0 \quad \tilde{y}_0]$	Ephemeris	Ephemeris	Vertices
`EphemCenter`	$[\varphi_1 \quad \varphi_2 \quad \dots \quad \varphi_{m_e} \quad \tilde{\alpha} \quad \tilde{x}_0 \quad \tilde{y}_0]$	Ephemeris	Ephemeris	Centers

A summary of the chosen test cases is shown in Table 3. These variants are divided into three general categories. First, we consider the strategy for selecting the moon perimeter; the `Full` and `FullCenter` sets compare the pixel-center and vertex methods. Both approaches fit all the model parameters. Next, we evaluate restricted formulations of the estimation problem. Here, the full formulation (`Full`) is compared to the raw initial estimates extracted directly from the images (`InitialEst`) and a simplified least-squares formulation that reduces the dimensionality of the fit by calculating the φ_i values from Eq. (72) at each iteration (`Restricted`). Finally, we consider the effect of predicting a and k from ephemeris. These values can be used as the initial estimates (`EphemEst`), or as constants that simplify the parameter fit (`EphemConst`, `EphemCenter`).

The system parameters for all of these tests are based on our baseline optical design and exposure level ($a = 30$, $E = 10$).

1. *Effect of Fitting Solution*

The moon–vector error for the first two sets of tests can be seen in Fig. 14. The error is quite low, even when using only the initial estimates. For most of the curves, the errors are highest when the moon is a thin crescent, that is, for small k, and lowest near the full moon. This is not unexpected because the moon most resembles a circle when it is full, and the algorithms rely less on the modeled shape. Poor performance at *low* values of k is unimportant because Fig. 5 suggests that good baffle design will minimize stray-light problems when $k < 0.2$. Even the initial guesses alone may be sufficiently accurate, depending on the mission's attitude knowledge requirements. The use of pixel centers as the perimeter points provides as much accuracy as using perimeter vertices. This strategy also reduces the number of perimeter points included in the fit, resulting in an appreciable reduction in computational requirements. Further computational savings can be attained using the `Restricted` dataset if some reduction in performance can be tolerated. This solution removes all the φ_i values from the least-squares fit and eliminates the need for a sparse matrix solver.

The error in the estimate of the sun vector is shown in Fig. 15. Comparing the magnitudes of σ_α and σ_s (from Fig. 14), it is clear that the moon–sun estimate will have the most effect on the attitude estimation accuracy. The errors, even for the most effective algorithms, are much higher than the satellite–moon measurements. The relative performance between the different solutions is similar to the σ_m case, but several interesting features are apparent. It is not clear why the `Restricted` solution outperforms the `Full` and `FullCenter` for crescent moon images (i.e., $k < 0.5$). We suspect that the difficulties arise from fitting the perimeter points near the cusps; these points could belong to either the semicircle or semi-ellipse curves. All the algorithms perform poorly at values of $k > 0.95$. In this regime, the moon looks almost full and loses the asymmetry that allows us to identify α. For a full moon, not only is it impossible to determine α from an overexposed moon image, but the attitude solution from the sun and moon vectors also becomes singular.

2. *Effect of Ephemeris Predictions*

The moon–vector error, using the ephemeris predictions for a and k, is shown in Fig. 16. When employed in place of the initial guesses, the predictions have little effect on performance, except at small values of k where the predictions actually degrade system performance. The vertical scale has been cropped to preserve the other features in the plot, but the maximum `EphemEst` error is about 1 deg. When used as constants, the predictions improve performance over the standard `Full` solution, but not dramatically. The choice of perimeter points has little effect on performance.

The performance impact of the ephemeris predictions on σ_s is pretty small (Fig. 17). The error is relatively consistent for $0.5 \leq k < 0.95$ and jumps considerably at higher values of k. One feature of note is the slightly elevated errors reported by the `EphemConst` solution between $0.1 \leq k < 0.35$. The exact mechanism for this anomalous behavior is not currently known, but it could be caused by

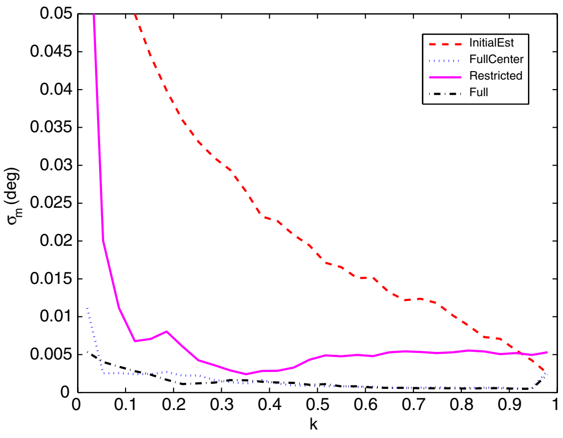

Fig. 14 Effect of mathematical formulation on satellite–moon estimate.

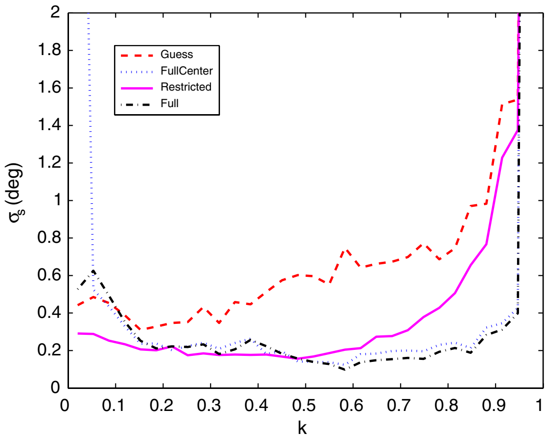

Fig. 15 Effect of mathematical formulation on satellite–sun estimate.

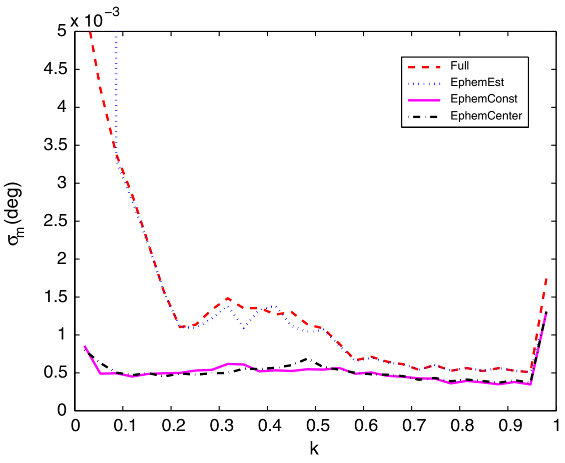

Fig. 16 Effect of ephemeris predictions on moon-vector estimation.

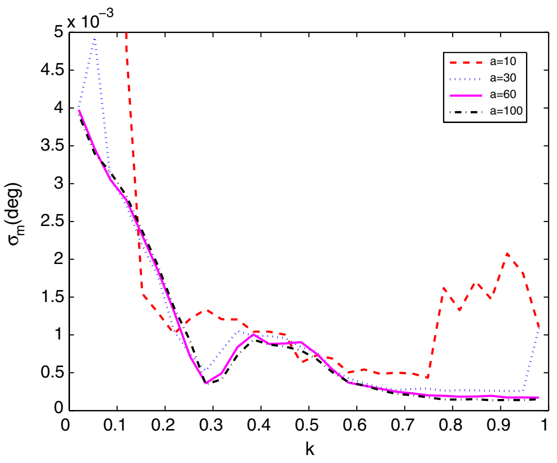

Fig. 18 Effect of camera resolution on moon-vector error.

overconstraining the boundary curves. Regardless of the processing approach, large errors near full-moon conditions remain apparent.

B. Effect of Optical Characteristics

The series of tests described in the previous section assume a baseline optical design. Different lens, detector, and orbit combinations will cause changes in a, and the length of the exposure time will alter the extent to which the image is overexposed (i.e., E). This section considers how changes in these parameters will affect system performance. The `Full` fit formulation was used throughout these two trials.

1. *Camera Resolution*

The value chosen for a conveniently abstracts away many of the details of the star-tracker optical design. Although a will change through the satellite orbit, even geostationary orbits will only see about a $\pm 10\%$ variation in moon radius. Hence, the sensor field of view and detector pixel count effectively fixes the apparent radius of the moon, which in turn determines the star-tracker resolution. The influence of this design parameter on σ_m is shown in Fig. 18. Even at rather low resolutions, for example, $a = 10$, the magnitude of the error is still very small in absolute terms. Although the errors grow quite high when k is small, this regime will have minimal operational impact. In contrast, we see from Fig. 19 that resolution has a significant effect on the performance-limiting σ_s. Improving the sensor resolution reduces σ_s, regardless of the lighting conditions. More important, with improved resolution, the moon-tracking algorithms remain effective at higher values of k. The extra pixels in the moon image permit the fitting algorithms to identify very small deviations from a perfect circle.

2. *Overexposure Effect*

Our approach to moon-tracking relies on deliberate overexposure of the moon disk images. This overexposure eliminates surface features on the face of the moon and simplifies the identification of the limb and terminator curves. One potential disadvantage of this approach is that prolonged exposure times could actually cause the edges in the moon image to become less distinct. Because of the common practice of defocusing star trackers (and diffraction limits to a lesser extent), pixels not directly in view of the moon may still be illuminated as the images are overexposed. To measure the effect of this phenomena on algorithm performance, we repeated some of the algorithm tests under different overexposure conditions. Because our simulation routines are not designed to replicate the craters and maria on the face of the moon, they cannot accurately produce images with E close to unity. This sets a lower limit on the range of E values that we can evaluate with our analysis.

The combined results from a number of trials are shown in Fig. 20. This figure shows both the σ_m and σ_s performance for the `Full` and `EphemCenter` formulations. These trials considered values of E on either side of the baseline value of $E = 10$. Overall, the effect on performance is slight. Some variability can be seen in σ_m performance for low k images in the `Full` trial. The data are inconclusive, but there may be an optimal value of E for processing this regime. That the variability is absent in the `EphemCenter` trial suggests that it is caused by interactions between the images and the `Full` algorithm formulation. When the σ_s results are examined, the `EphemCenter` results show variability, particularly in the range $0.1 \leq k < 0.4$. In this range, performance decreases with increasing E. The moon's shape and size is constant in these tests; only position and orientation are allowed to vary. Thus, the performance drop may

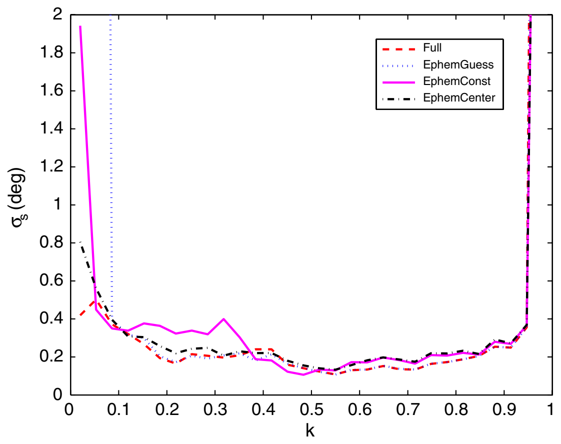

Fig. 17 Effect of ephemeris on sun-vector estimation.

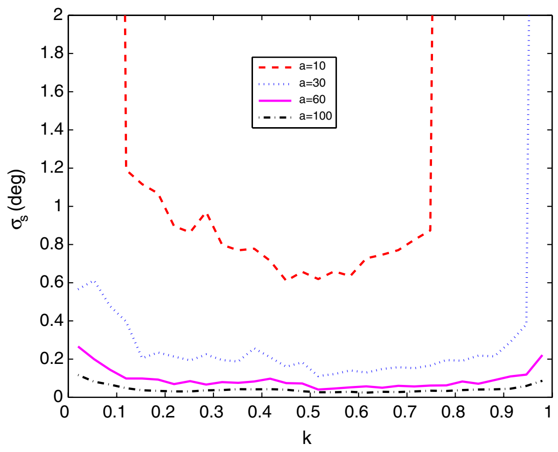

Fig. 19 Effect of camera resolution on sun-vector error.

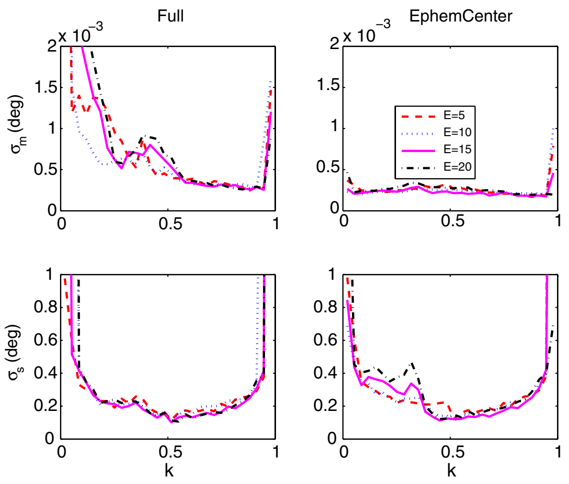

Fig. 20 Effect of overexposure on system performance.

be caused by distortions to the concave moon shape, particularly near the cusps where the two boundary curves lie close together. At lower values of k, the performance is driven by the inherent difficulty in registering and interpreting very narrow crescents.

VI. Conclusions

In this paper, we have considered the feasibility of implementing a fallback, moon-tracking operating mode for microsatellite star trackers when the moon is in the field of view. During these conditions, illumination from the moon may introduce excess light into the star tracker making conventional star detection difficult. Because the shape and position of the partially illuminated moon contains enough information to compute measurement vectors to the moon and sun, moon-tracking measurements can be combined with ephemeris predictions to allow a full attitude solution under most conditions. A three-axis attitude fix during moon incursions is a significant step toward establishing the feasibility of a star-tracker-only attitude estimation system.

Practical deployment of moon-tracking algorithms must consider the operational context of the moon-tracking problem. An examination of the visible magnitude of the moon as function of its phase revealed that, for slim crescent moons, stray light would not likely interfere with detecting magnitude 4.0 stars. The visual magnitude threshold and the limiting crescent moon size will be influenced by the star tracker's baffle effectiveness, but there exists a definite lower bound on when moon tracking will be necessary.

To test our moon-tracking algorithms, we evaluated the accuracy of parameter estimation using several different algorithm variants. In all cases, moon position estimates within the image were very good. The corresponding errors in the moon-vector estimates were on the order of 10^{-3} deg. Even the raw estimates extracted from the moon images were accurate to within about 0.02 deg. If a two-axis fix is sufficient during a short moon incursion, the initial estimates alone may suffice. Three-axis performance was limited by the accuracy of the moon rotation angle. Although some of the algorithms were able to maintain 0.25 deg accuracy over 85% of moon conditions, all algorithms had difficulty when the moon was nearly full. Further algorithm refinement (e.g., weighted least squares) may extend the operational envelope of our moon-tracking routines, but a perfect full moon represents a degenerate solution in any case. If a three-axis solution must be maintained through such an incursion, a different attitude estimation scheme must be adopted; Park and Mortari [2] have suggested tracking craters or other distinctive surface features. A detailed examination of the operational requirements of any particular mission may be necessary to tradeoff between algorithmic complexity and availability.

The choice of processing scheme had some notable effects on performance. Using pixel centers as perimeter points demonstrated clear advantages over pixel vertices. These algorithms improved accuracy and reduced the dimensionality of the numerical fit. The algorithms that calculated lunar size and phase generated from ephemeris predictions generally outperformed algorithms estimating these quantities from the moon images. In some scenarios (relative attitude acquisition in the absence of reliable ephemerides, for instance), extracting these quantities from the images could be valuable, but these applications are very narrow.

In our tests, improving the star-tracker angular resolution yielded improvements in mean performance and increased algorithm tolerance to images taken when the moon is nearly full. The abstraction of the optical and detector design into a single parameter, although useful for this study, masks several important considerations. For instance, improving the inherent star-tracker resolution, that is, deg/pixel, can be accomplished by using a detector with smaller pixels or by optically narrowing the sensor field of view. Narrow FOV sensors must rely on dimmer stars for their normal attitude fix and may experience stray-light problems when the moon is at a dimmer visual magnitude. Consequently, the operational threshold where the moon-tracking mode is needed will move toward slimmer crescent moon shapes. On the other hand, improving resolution by increasing the detector pixel density may benefit the moon-tracking performance, but detector noise may adversely affect nominal star-tracking capabilities. Therefore, any redesign aimed at improving sensor accuracy must consider the performance implications in all operational modes.

Our algorithms, although demonstrably promising in simulation, would certainly benefit from experimental validation. Even without ground truth for the orientation estimates, imaging the moon under a wide range of illumination conditions will lend credence to our techniques. In particular, a long field trial will establish that the image saturation can be maintained through simple control logic. Our tests indicate that performance is only weakly tied to the level of exposure, and so maintaining precise exposure control is probably unnecessary. A series of unverified tests are planned for the near future, and we are also investigating opportunities at local observatories to conduct verifiable trials.

Throughout this study, we have consciously focused our attention on improving the accuracy of individual measurements and expanding the operational envelope for which the algorithms work. In practice, individual measurements of attitude are almost always combined with dynamics propagation and recursive state estimation using Kalman filtering or related techniques. The ultimate suitability of moon tracking using star trackers depends greatly on the specific requirements of a candidate mission. Because neither the moon phase nor position are stochastic, careful mission planning may be able to avoid or minimize the duration of full-moon incursions during nominal operations when good attitude knowledge must be maintained. Conversely, during idle, or safe-mode, operations, a good lock on the sun is often the first priority. Even during a full moon, our moon-tracking routines will still provide a high-quality body-referenced sun vector. Thus, we feel that moon tracking can enhance the utility of star trackers and may ultimately enable simpler and more effective micro- and nanosatellite hardware designs.

Acknowledgment

The author wishes to thank Doug Sinclair from Sinclair Interplanetary for technical assistance and donation of the prototype star tracker camera.

References

[1] Mortari, D., "Moon-Sun Attitude Sensor," *Journal of Spacecraft and Rockets*, Vol. 34, No. 3, 1997, pp. 360–364. doi:10.2514/2.3217

[2] Park, K. J., and Mortari, D., "Planet or Moon Image Processing for Spacecraft Attitude Estimation," *Proceedings of the AAS/AIAA Space Flight Mechanics Meeting*, American Astronautical Society Paper AAS 06-109, 2006.

[3] Pingyuan, C., Fuzhan, Y., and Hutao, C., "Attitude and Position Determination Scheme of Lunar Rovers Basing on the Celestial Vectors

Observation," *IEEE International Conference on Integration Technology*, Inst. of Electrical and Electronics Engineers, New York, 2007, pp. 538–543.

[4] Meeus, J. H., *Astronomical Algorithms*, Willmann–Bell, 1991, pp. 163–176, 337–344, 349–354.

[5] Leinert, C., and Klueppelberg, D., "Stray Light Suppression in Optical Space Experiments," *Applied Optics*, Vol. 13, No. 3, 1974, pp. 556–564.
doi:10.1364/AO.13.000556

[6] Jorgensen, J. L., "In-Orbit Performance of a Fully Autonomous Star Tracker," *Spacecraft Guidance, Navigation and Control Systems*, B. Schürmann (ed.), Vol. 425, ESA SP-425, 2000, pp. 103–110.

[7] Laher, R., Catanzarite, J., Conrow, T., Correll, T., Chen, R., Everett, D., Shupe, D., Lonsdale, C., Hacking, P., and Gautier, N., "Attitude Control System and Star Tracker Performance of the Wide-Field Infrared Explorer Spacecraft," *AAS/AIAA Spaceight Mechanics Meeting*, American Astronautical Society, Springfield, VA, 2000.

[8] Wertz, J. R., *Spacecraft Attitude Determination and Control*, Kluwer Academic, Norwell, MA, 1978, p. 823.

[9] Crassidis, J. L., Markley, F. L., and Cheng, Y., "Survey of Nonlinear Attitude Estimation Methods," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 1, 2007, pp. 12–28.
doi:10.2514/1.22452

[10] Goreaud, F., and Pélissier, R., "On Explicit Formulas of Edge Effect Correction for Ripley's *k*-Function," *Journal of Vegetation Science: Official Organ of the International Association for Vegetation Science*, Vol. 10, No. 3, 1999, pp. 433–438.
doi:10.2307/3237072

[11] Healey, G., and Kondepudy, R., "Radiometric CCD Camera Calibration and Noise Estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 3, 1994, pp. 267–276.
doi:10.1109/34.276126

[12] Sylvester, J. J., "A Question in the Geometry of Situation," *Quarterly Journal of Pure and Applied Mathematics*, Vol. 1, 1857, p. 79.

[13] Hearn, D. W., and Vijay, J., "Efficient Algorithms for the (Weighted) Minimum Circle Problem," *Operations Research*, Vol. 30, No. 4, 1982, pp. 777–795.
doi:10.1287/opre.30.4.777

[14] Oommen, B. J., "An Efficient Geometric Solution to the Minimum Spanning Circle Problem," *Operations Research* [online], Vol. 35, No. 1, 1987, pp. 80–86; http://www.jstor.org/stable/170912.

[15] Gander, W., Golub, G. H., and Strebel, R., "Least-Squares Fitting of Circles and Ellipses," *BIT Numerical Mathematics* [online], Vol. 34, No. 4, Dec. 1994, pp. 558–578; http://dx.doi.org/10.1007/BF01934268.