SSC07-VIII-5

Algorithm Enhancements for the SS-411 Digital Sun Sensor

John Enright
Department of Aerospace Engineering, Ryerson University
350 Victoria St., Toronto, ON, M5B2K3 CANADA, 416-979-5000 (x4174)
jenright@ryerson.ca

Doug Sinclair
Sinclair Interplanetary
268 Claremont St., Toronto, ON, CANADA, M6J 2N3, 647-286-3761
dns@sinclairinterplanetary.com

ABSTRACT

This paper discusses the evolution of the SS-411 series digital sun sensor. The earlier SS-256 and SS-330 models have proven themselves in orbit on low cost satellite missions. The SS-411 represents a further hardware revision with enhanced robustness and improved attitude estimation performance. To complement the latest hardware improvements, researchers in the Space Avionics and Instrumentation Laboratory (SAIL) at Ryerson University in Toronto have developed advanced signal processing routines compatible with the SS-411. These routines significantly improve the accuracy of the sensor's estimation without increasing manufacturing complexity. With this advanced processing, the sensor maintains a mean accuracy of 0.16° over the entire $\pm 70°$ field of view.

INTRODUCTION

As the microsatellite industry matures, the competing demands for performance and simplicity create niches for new devices. The Sinclair Interplanetary SS-411 digital sun sensor (Figure 1) represents the latest revision in a series of small and capable microsatellite sensors. This study details the latest changes to the sensor design with particular focus on a set of new signal processing algorithms.

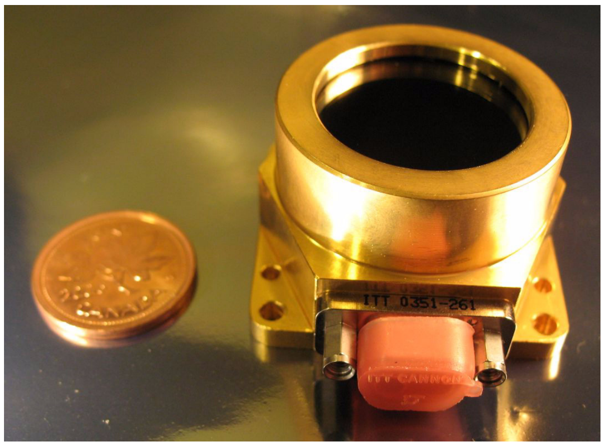

Figure 1: SS-411 digital sun sensor with penny for scale.

The sensor employs narrow slits to create a series of bright peaks on a linear detector array. The accuracy in the output sun vector is directly related to the accuracy of the *peak position estimate (PPE).* Previous laboratory and simulation studies [1][2] have post-processed images from the SS-256 and SS-330 model sensors to localize the peaks to a small fraction of a pixel, and thus produce very precise sun-estimates. This study details the adaptation of these algorithms from off-line use to embedded implementation on the SS-411's internal 8-bit processor.

The SS-411 design.

Analog two-axis sun sensors are commonly made from four-quadrant photodiodes. The sun angles can be determined by a simple function of the four output currents. However, these sensors are vulnerable to incoming light from non-solar sources: the earth, the moon, or glint from spacecraft appendages. This unwanted light cannot be distinguished from sunlight and so it decreases the sensor's operational accuracy.

Digital sun sensors use an array of many photosensitive pixels. By increasing the number of scalar measurements it becomes possible to resolve multiple light sources and to discard those that are not the sun. Many two-axis digital sun sensors are available. Some use a two-dimensional detector array such as a CCD or active pixel sensor [3][4]. Others use two orthogonal linear detector arrays[5]. The Sinclair Interplanetary sensors are unusual in that they use only a single linear detector but are able to determine the sun location in two axes.

Incoming sunlight passes through a metal mask in which four thin slits have been cut (Figure 2). An image of this pattern is projected onto a linear detector array, creating a one-dimensional brightness profile containing four strong peaks. One pair of peaks is close together, while the other pair has a wider spacing (Figure 3). This allows a computer algorithm contained within the sensor to distinguish between them and to locate the two-dimensional offset between the mask and the detector. A second algorithm models the sensor geometry including refractive effects and produces the final output: a vector from the sensor to the sun.

Current Design

The sensor is built from high-end COTS components. At its heart is a 16mm long linear detector array with 256 pixels. The detector has a fast electronic shutter, but even at the maximum speed the pixels can saturate in direct sunlight. An optical filter with a transmission of 10% or less attenuates the incoming light and allows reasonable shutter speeds. All of the optical parts use the industry standard 25mm diameter circular form factor.

A concern early in concept development was how to vent the sensor without letting stray light strike the detector. Light-tight venting filters are available, but present mechanical integration concerns. Instead, the problem of venting was avoided by designing the device so that it contains no air. The optical elements are bonded directly to each other and to the detector using transparent adhesive. The rest of the unused internal volume is completely filled with opaque potting compound.

The detector is mounted to a printed circuit board that also contains a microcontroller (Figure 4). An on-board ADC digitizes the detector output and the CPU executes the image processing algorithms. A serial bus connects the microcontroller to the satellite's ACS computer. Various different bus technologies are supported: asynchronous, SPI, I2C, and CANbus. The sensor produces digital sun position vectors, as well as exposure and temperature telemetry. It accepts synchronization and reconfiguration commands, or even software patches while on-orbit.

Heritage

The first Sinclair Interplanetary sun sensor design was designated the SS-256. It was followed in succeeding years by the SS-330, and recently the SS-411 (Figure 5).

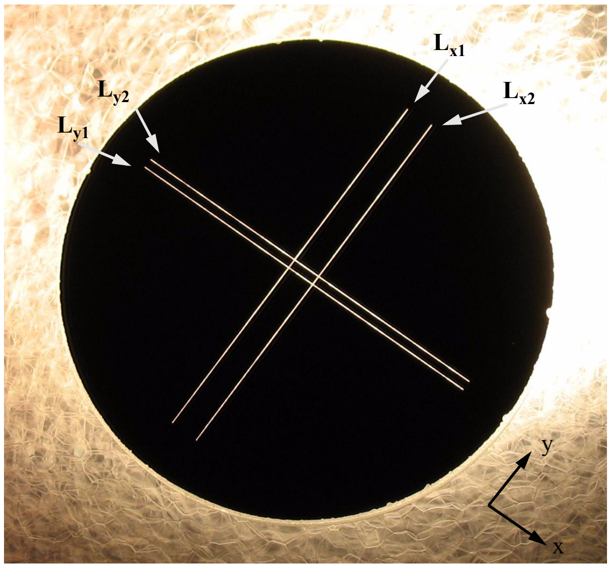

Figure 2: The SS-411 Mask, backlit to show slits. The mask is 25mm in diameter.

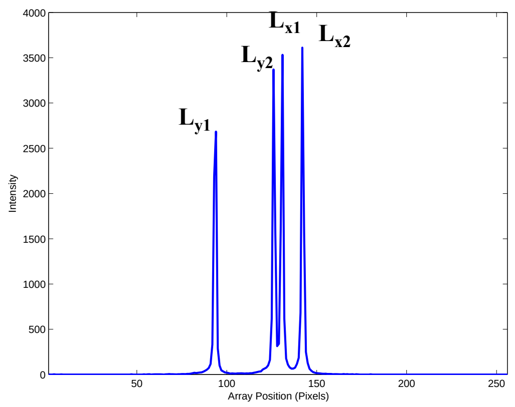

Figure 3: Sample Images from the SS-411. The dotted peaks have been labelled with their generating slits.

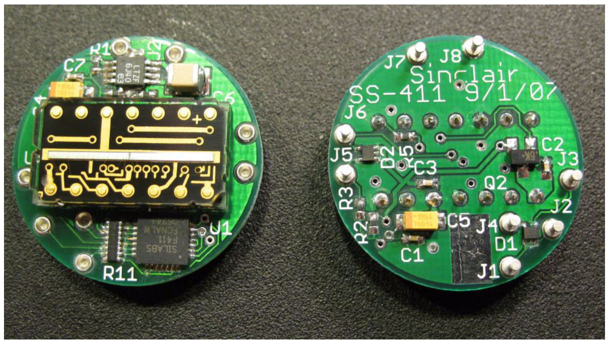

Figure 4: The SS-411 Electronics. The detector and microcontroller can be seen in the left figure.

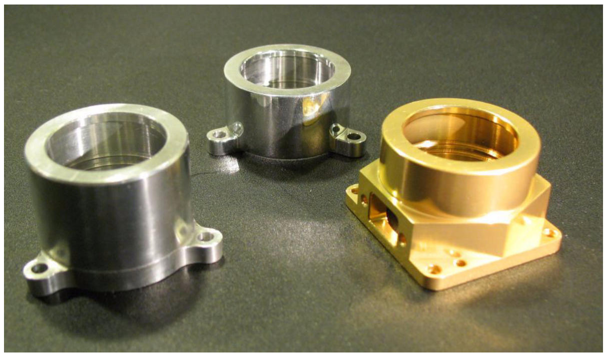

Figure 5: Sensor housings: SS-256 (left), SS-330 (center) and SS-411 (right)

All three sensors employ the same basic principle but numerous refinements have been made to improve performance and usability. Table 1 summarizes the key features of each design.

The SS-256 used an optical design with three bonded glass elements: a clear mask substrate and two stacked neutral density filters. The electronics occupied two discrete circuit boards stacked into a "wedding cake" configuration. Despite its large size the processor was very modest. With only 384 bytes of RAM the device was limited to 8-bit quantization and simple fixed-point algorithms

TABLE 1: Sensor Evolution

	SS-256	SS-330	SS-411
History			
First Production	2004	2005	2007
# on-orbit	5	9*	0
Mechanical			
Height	22 mm	17 mm	21 mm
Interface	Flywires	Flywires	Micro-D
Surface	Aluminum	Aluminum	Gold
Optical			
Front Window	Glass	Sapphire	Sapphire
# of elements	3	3	2
Electrical			
Circuit Boards	2	1	1
CPU clock	8 MHz	24.5 MHz	49 MHz
RAM	0.375 kB	0.75 kB	2.25 kB
ROM	7.5 kB	7.5 kB	31.5 kB
Supply Voltage	3.3 V	<16 V	<50 V

* 5 on-orbit, presently; 4 more expected by publication

The SS-330 upgrade replaced the glass front window with sapphire for thermal and mechanical durability. The electronics were compressed into a single PCB, easing the assembly process and shrinking the enclosure height by 5 mm. A more powerful processor allowed floating-point computations and complex packet protocols. An on-board linear regulator accepted input power up to 16 V, allowing direct connection to many microsatellite busses.

The most recent step is the SS-411. The mask is deposited directly onto the back surface of the sapphire window, eliminating the glass substrate and simplifying the optical stack. A new mechanical design mounts a micro-D connector eliminating the fragile flywires but adding 4 mm to the height. The case is plated in gold for environmental and thermal protection. Another processor upgrade permits the use of sophisticated image processing algorithms in real-time. The linear regulator is also upgraded, now allowing direct connection to +28 V avionics power.

The first fifteen SS-411 units have been delivered to customers, and first flight is expected in the coming year.

IMAGE PROCESSING

The SS-411 and its predecessors produce images from their pixel arrays. Turning this array of intensity readings, $I[n]$, into an estimate of the sun location requires several processing steps. The position of each bright peak must be coarsely located and associated with one or more contributing slits. As the sun moves in the field of view it creates distinct shifts (m_x and m_y) in the peak patterns on each axis. Estimating these quantities is called *image segmentation*. These estimates must be refined to a fraction of a pixel in a process of *peak position estimation*. Finally, the precise peak locations must be used to calculate the sun vector, $\tilde{s}_0$. This process is shown graphically in Figure 6.

Sometimes we are interested in the estimated locations of each individual peak in the image (i.e. $\tilde{m}_{x_1}, \tilde{m}_{x_2}, \tilde{m}_{y_1}, \tilde{m}_{y_2}$). Other times it is only the mean position ($\tilde{m}_x, \tilde{m}_y$) of a pair of peaks that is important. These quantities are easily related:

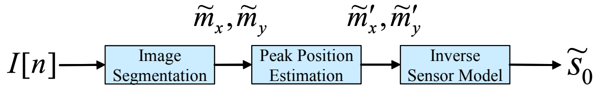

Figure 6: Image processing steps.

$$\tilde{m}_x = (\tilde{m}_{x_1} + \tilde{m}_{x_2})/2 \quad (1)$$

$$\tilde{m}_{x_{1/2}} = \tilde{m}_x \pm L_x/\Delta X \quad (2)$$

similarly

$$\tilde{m}_y = (\tilde{m}_{y_1} + \tilde{m}_{y_2})/2 \quad (3)$$

$$\tilde{m}_{y_{1/2}} = \tilde{m}_y \pm L_y/\Delta X \quad (4)$$

where the pairs of peaks are spaced $2L_x$, and $2L_y$ apart and the spacing between pixels is ΔX

Coarse Identification and Classification

The first task of the image processing sequence is to identify the rough location and source of each peak in the sensor image. It is helpful to introduce some nomenclature. Each aperture slit creates a bright line on the focal plane. These lines are *logical* peaks. We will refer to each bright spike in the image as a *physical* peak. The physical peaks are caused by the logical peaks where they intersect the detector, but since the illumination from the two axes can overlap, we do not always see four distinct physical peaks in the image. Several factors further complicate the identification process. The amplitude of each logical peak varies as the sun moves in the field of view. This requires care in setting absolute thresholds for peak intensities. Noise can corrupt the sensor image, and we must be able to differentiate between noise-spikes and actual peaks. Two sample sensor images and peak mapping is shown in Figure 7.

Our first task is to identify the physical peaks in the image. We implement this search with a simple state-machine pattern matching algorithm. To identify a physical peak at pixel-k, the image must satisfy the following criteria:

- The image intensity at k (i.e. $I[k]$) must be greater than a minimum threshold (typically 25% of the image peak intensity.
- The image intensity at k must be the greatest intensity from the start of the image or the last dip between peaks. A dip is identified when we drop at least 33% from the previous peak.

The threshold levels described above are empirical factors chosen to provide good performance over a large number of trial cases. They are deliberately set to be slightly conservative and will sometimes identify a single physical peak where multiple peaks can be discerned by eye. Subsequent refinement of the peak locations correct for most inaccuracies that may be introduced by this process.

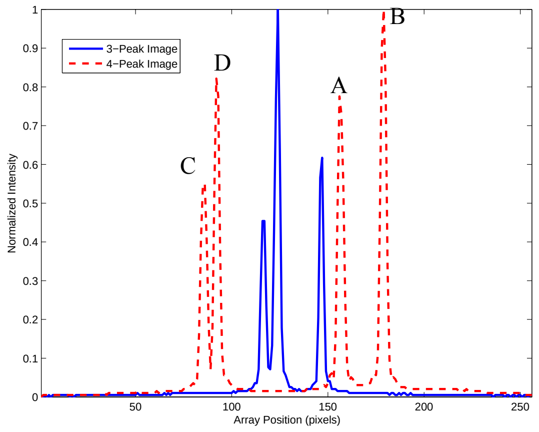

Figure 7: Two sample sensor images. One image is annotated to show logical associations to physical peaks

Once the physical peak locations (p_i) have been determined, we assign logical peaks to the physical peaks by looking for the best match to the known peak spacings. I.e. we attempt to find a set of assignments from the p_i's to the $\tilde{m}$'s that minimizes:

$$\Phi = (2L_x/\Delta X - (\tilde{m}_{x_2} - \tilde{m}_{x_1}))^2 + (2L_y/\Delta X - (\tilde{m}_{y_2} - \tilde{m}_{y_1}))^2 \quad (5)$$

In practical terms we can explicitly restrict the number of cases that we consider. There are five cases to evaluate when we find four distinct physical peaks. When we only find three peaks, there are four cases to consider.

Peak Position Estimation

Illumination from a single slit extends over several pixels. Intuitively this suggests that there is more information to be gleaned from an image than merely the indices of the brightest pixels. We recognize that attitude information is spread over a number of pixel readings, and that this information should reflect an underlying structure (i.e. the peaks occur at known spacing). Thus, the knowledge of what we *expect* to see in the image allows us to refine our analysis of what we do see.

We formulate our expectations of the image appearance into a parameterized, analytic model. The parameters of the model capture quantities that we wish to measure (e.g., $\tilde{m}_x, \tilde{m}_y$) as well as any expected image-to-image variations (e.g. peak amplitude). We can then use popular minimization strategies to choose the set of parameters that best matches the observed image. For the SS-411, we have adapted the non-linear least squares (NLSQ) algorithm. NLSQ is a generalization of linear least-squares curve fitting that permits more complicated model parameterizations. One downside is that the algorithm becomes iterative rather than deterministic.

We denote the modeled, continuous pattern from a single slit as $I_{mdl_1}(\tau)$; a function of the position τ on the array. Thus, we expect that the net illumination at the detector will be of the form:

$$I_{mdl}(\tau) = a_1 A(\tau) + a_2 B(\tau) + b_1 C(\tau) + b_2 D(\tau) \tag{6}$$

where

$$\begin{aligned} A &= I_{mdl_1}(\tau - \tilde{m}_{x_1}) \\ B &= I_{mdl_1}(\tau - \tilde{m}_{x_1} - 2L_x/\Delta X) \\ C &= I_{mdl_1}(\tau - \tilde{m}_{y_1}) \\ D &= I_{mdl_1}(\tau - \tilde{m}_{y_1} - 2L_y/\Delta X) \end{aligned} \tag{7}$$

This model contains six parameters that must be identified ($a_1, a_2, b_1, b_2, \tilde{m}_{x_1}, \tilde{m}_{y_1}$); four amplitudes, and two displacements. Although we are only truly interested in finding the latter two quantities, since the amplitudes do vary, they must be identified as well.

For notational convenience, we express the image components and model parameter estimates as vectors:

$$\boldsymbol{A} = \{A_n\} = \left[A(\tau_1)\ A(\tau_2)\ \ldots\right]^T \text{ (similarly with } \boldsymbol{B}\text{, etc.)} \tag{8}$$

as well as the set of model parameters:

$$\lambda = \left[a_1\ a_2\ b_1\ b_2\ \tilde{m}_{x_1}\ \tilde{m}_{y_1}\right]^T \tag{9}$$

Following a typical NLSQ implementation, we define the error observed, $d\beta_n$, between a real image ($I_s[i]$) and a modeled image:

$$d\beta_n = I_s[n] - I_{mdl}(\tau_n, a_1, a_2, b_1, b_2, \tilde{m}_{x_1}, \tilde{m}_{y_1}) \tag{10}$$

Now we introduce the matrix, $\boldsymbol{Q}$, which is formed from the partial derivatives of I_{mdl}, with respect to the model parameters:

$$\boldsymbol{Q} = \begin{bmatrix} \left.\dfrac{\partial I_{mdl}}{\partial a_1}\right|_{\tau_1} & \left.\dfrac{\partial I_{mdl}}{\partial a_2}\right|_{\tau_1} & \cdots \\ \left.\dfrac{\partial I_{mdl}}{\partial a_1}\right|_{\tau_2} & \ddots & \\ \vdots & & \left.\dfrac{\partial I_{mdl}}{\partial \tilde{m}_{y_1}}\right|_{\tau_{256}} \end{bmatrix} \tag{11}$$

Each element in this matrix represents a partial derivative evaluated at the current estimate of the model parameters, and a particular τ_i. If we define the component partial derivatives of I_{mdl_1} :

$$\begin{aligned} E(\tau) &= -\frac{\partial}{\partial u}(I_{mdl_1}(u))\Big|_{u = \tau - m_{x_1}} \\ F(\tau) &= -\frac{\partial}{\partial u}(I_{mdl_1}(u))\Big|_{u = \tau - m_{x_1} - 2L_x/\Delta X} \\ G(\tau) &= -\frac{\partial}{\partial u}(I_{mdl_1}(u))\Big|_{u = \tau - m_{y_1}} \\ H(\tau) &= -\frac{\partial}{\partial u}(I_{mdl_1}(u))\Big|_{u = \tau - m_{y_1} - 2L_y/\Delta X} \end{aligned} \tag{12}$$

then, $\boldsymbol{Q}$ can be written in matrix form, as a series of column vectors:

$$\boldsymbol{Q} = \left[\boldsymbol{A}\ \boldsymbol{B}\ \boldsymbol{C}\ \boldsymbol{D}\ (a_1\boldsymbol{E} + a_2\boldsymbol{F})\ (b_1\boldsymbol{G} + b_2\boldsymbol{H})\right]$$

We can show that for each iteration of the algorithm, the parameter update, $\boldsymbol{d}\lambda$, is found by solving the linear system:

$$\boldsymbol{Q}^T\boldsymbol{d}\beta = (\boldsymbol{Q}^T\boldsymbol{Q})\boldsymbol{d}\lambda \tag{13}$$

This can be expressed a little more succinctly if we make the substitutions $\boldsymbol{S} = \boldsymbol{Q}^T\boldsymbol{Q}$, and $\boldsymbol{b} = \boldsymbol{Q}^T\boldsymbol{d}\beta$. This gives:

$$\boldsymbol{S}\boldsymbol{d}\lambda = \boldsymbol{b} \tag{14}$$

For our standard model the system has six equations and six unknowns. The algorithm terminates after a set number of iterations, or when the changes to λ drop below a certain tolerance. Although the latter makes more sense from a numerical standpoint, the former exit criterion is a better match to embedded applications where temporal determinism is desired.

The system of equations in (14) is well behaved when there are four distinct physical peaks in the image. When illumination from different slits overlap, the linear system for the parameter update becomes ill-conditioned. Essentially, the there is no way of separating the two arbitrary overlapping peak amplitudes. In order to resolve this, three-peak images must be treated as a series of special cases; one parameter is removed and the overlapping peaks are assumed to have the same amplitude. Table 2 enumerates the different types of images that can be encountered, together with the corresponding $\boldsymbol{Q}$ matrices.

Optics Model

The last computational step in image processing is to take the refined estimates of peak locations and use them, together with knowledge of the sensor optics to produce a sun vector estimate. Sun vector estimation is a direct consequence of the image formation process. The simplicity of the optical design allows us to capture essential behaviours through simple geometric analysis.

TABLE 2: Image classifications

Image Type	Characteristics	Q Matrix
0	Any image with four physical peaks	$\left[\boldsymbol{A}\ \boldsymbol{B}\ \boldsymbol{C}\ \boldsymbol{D}\ (a_1\boldsymbol{E}+a_2\boldsymbol{F})\ (b_1\boldsymbol{G}+b_2\boldsymbol{H})\right]$
1	A and C overlap	$\left[(\boldsymbol{A}+\boldsymbol{C})\ \boldsymbol{B}\ \boldsymbol{D}\ (a_1\boldsymbol{E}+a_2\boldsymbol{F})\ (a_1\boldsymbol{G}+b_2\boldsymbol{H})\right]$
2	B and C overlap	$\left[\boldsymbol{A}\ (\boldsymbol{B}+\boldsymbol{C})\ \boldsymbol{D}\ (a_1\boldsymbol{E}+a_2\boldsymbol{F})\ (b_2\boldsymbol{G}+b_2\boldsymbol{H})\right]$
3	A and D overlap	$\left[(\boldsymbol{A}+\boldsymbol{D})\ \boldsymbol{B}\ \boldsymbol{C}\ (a_1\boldsymbol{E}+a_2\boldsymbol{F})\ (b_1\boldsymbol{G}+a_1\boldsymbol{H})\right]$
4	B and D overlap	$\left[\boldsymbol{A}\ (\boldsymbol{B}+\boldsymbol{D})\ \boldsymbol{C}\ (a_1\boldsymbol{E}+a_2\boldsymbol{F})\ (a_2\boldsymbol{G}+b_2\boldsymbol{H})\right]$

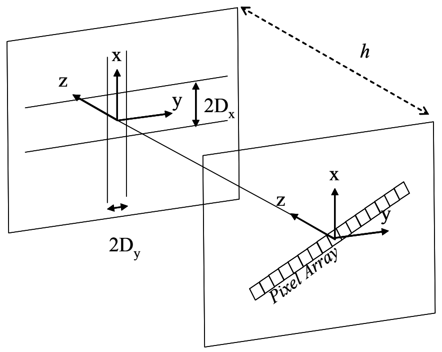

Figure 8: Geometric model of sensor showing aperture and image planes.

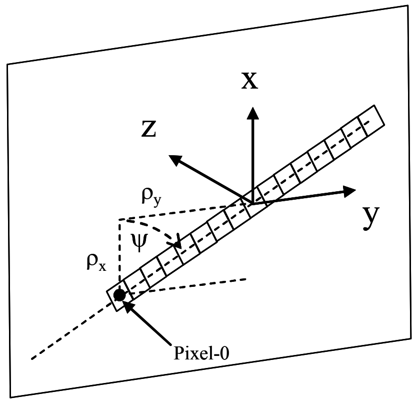

Figure 9: The model must locate and orient the sensor array in the image plane

A schematic of the sensor optics is shown in Figure 8. The front mask on the sensors allows light to enter, refracting through the filter separating the mask from the image plane. Each slit produces a bright line of illumination on the image plane. The placement of these bright lines depends on the mask geometry, captured in D_x and D_y, the filter thickness, h, and the sun vector $\hat{s}_0$.

Not shown on the diagram, but important to the optical model is a z-axis rotation through an angle ϕ. This angle measures the twist between the *external sensor frame* (fixed to the housing), and the *internal sensor frame* (fixed to the slits).

The lines of illumination create regions of high response where they intersect the pixel array. The position and orientation of the detector will determine which pixels will be lit. Array placement can be captured by three simple parameters: the coordinates of the first pixel, (ρ_x, ρ_y), and angle between the array and the x-axis, ψ.

Using simple vector geometry it is possible to show that the components, $\left[\tilde{s}_x\ \tilde{s}_y\ \tilde{s}_z\right]$, of the refracted inward-pointing sun vector, $\tilde{s}$, are given by:

$$\begin{aligned}
\tilde{s}_x &= -\frac{s_z}{h}\left[\Delta X\cos\psi\left(\frac{\tilde{m}_{x_1}+\tilde{m}_{x_2}}{2}\right)+\rho_x\right]\\
\tilde{s}_y &= -\frac{s_z}{h}\left[\Delta X\sin\psi\left(\frac{\tilde{m}_{y_1}+\tilde{m}_{y_2}}{2}\right)+\rho_y\right]\\
\tilde{s}_z &= -\sqrt{1-s_x^2-s_y^2}
\end{aligned}\tag{15}$$

Accounting for the index of refraction of the filter (n_{glass}), and rotating the vector into the external frame gives a set of expressions for the estimated vector *to* the sun:

$$\begin{aligned}
\tilde{s}_{0_x} &= -n_{glass}(\cos\phi\tilde{s}_x+\sin\phi\tilde{s}_y)\\
\tilde{s}_{0_y} &= -n_{glass}(-\sin\phi\tilde{s}_x+\cos\phi\tilde{s}_y)\\
\tilde{s}_{0_z} &= \sqrt{1-\tilde{s}_{0_x}^2-\tilde{s}_{0_y}^2}
\end{aligned}\tag{16}$$

It is worth mentioning that the calculations necessary to calculate $\tilde{s}_0$ are quite simple. If the transcendental evaluations and $1/h$ are computed off-line, calculating $\tilde{s}_0$ requires only adds, multiplies and a single square-root.

EMBEDDED SIGNAL PROCESSING

Embedding the signal processing on-board the SS-411 microcontroller was not trivial, but a disciplined approach facilitated the process. Our approach combined knowledge of the target architecture, judicious planning, hand optimization, and some well-chosen approximations.

Processing on the SS-411

The processor on the SS-411 sun-sensor is a Silicon Labs C8051F411 microcontroller [6]. Although this processor is capable of about 50 MIPS, it offers no native floating-point support and has only 2304 *bytes* of RAM. Of this, 512 bytes must be used to store the sensor image itself. This poses several challenges when re-engineering the sensor algorithms for embedded implementation. Processing speed is the first concern. While the processor does allow floating point arithmetic, these operations must be synthesized by the compiler from single byte manipulations. The fast clock speed made matters a little easier, but we still wished to eliminate extraneous calculations and use fixed-point arithmetic where possible. The memory restrictions were much more demanding. Naively calculating the entire Q-matrix would require 1536 elements and at least twice that number of bytes. That alone drove home the need to carefully consider the order and extent of the calculations performed during each iteration.

Migration Strategy

For most of SAIL's previous algorithm research we processed our sensor data off-line. Images were transferred into Matlab and examined in that environment. Matlab provides an excellent research environment but mathematical operations are typically performed at a much higher conceptual level than would be available on the C8051. When formulating a testing plan for this project, we did not want to abandon the convenient facilities for regression testing that Matlab offers. After some consideration, we devised a strategy for migration from Matlab to the C8051 that made the best use of our available resources (Figure 10).

Starting from the existing processing routines, the first step in the algorithm migration was to reduce the use of Matlab processing idioms and adopt a more C-like implementation. Free form array manipulation and vectorized operations were replaced by explicit loops. All processing was performed in exactly the same way as would be done on the sensor. All high-level operations had to be replaced by sequences of simple steps. As an example, an LU-Decomposition algorithm was implemented explicitly to solve the linear system encountered in each iteration.

The next step was to recode the Matlab routines into C. Most of the major algorithmic changes were performed in the previous step, so this process created few problems. The image segmentation, PPE and optics processing routines were implemented as a set of C-functions. With a small amount of wrapper code, we could provide different interfaces to the processing routines. This gave rise to two variations. A stand-alone version of the algorithms, useful for detailed debugging, and a simple 'Matlab-MEX' style wrapper. This latter version allowed the C-routines to be invoked from Matlab test scripts; a feature essential for regression testing.

After the initial implementation, subsequent performance optimization was performed at the C-level. We did not update the simplified Matlab processing routines. This was justifiable since we could still compare algorithm results to the previous step in the chain.

Once the C-routines were working satisfactorily, we cross-compiled the code for the C8051. We purchased a C8051 development kit (DK) from Silicon Labs to enable software development before the flight hardware was complete. This kit contained a processor and various peripherals including a stop-mode debugger interface and a serial-port for external communication. The C8051 has a distinctive memory-map and the compiler supported various directives that would ensure that variables ended up in appropriate memory blocks. Some optimization and code changes were necessary to ensure that the most frequently accessed variables were located in the fastest types of memory.

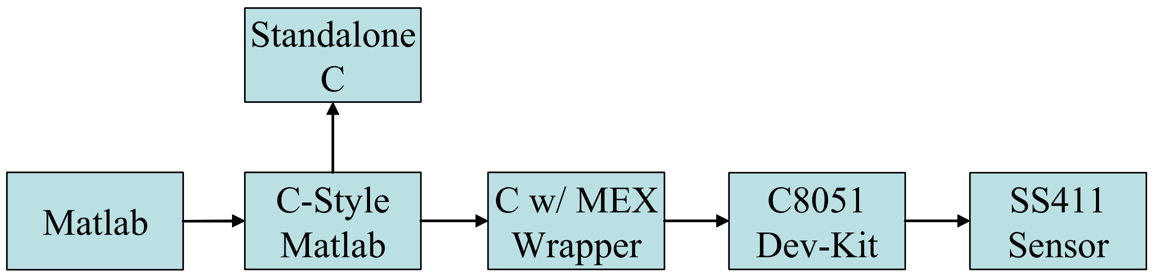

Figure 10: Development Strategy. All test data returned to Matlab for comparison to previous tests.

At each stage in this migration the algorithm performance was validated against a set of 800 test images. Performing this regression testing assured us that the consistent algorithm performance could be traced back to the trusted Matlab implementation. Any large variations indicated problems with the implementation. In the end, there were some small variations in the off-line and on-line estimates due to a change from double to single precision arithmetic, but these effects were negligible.

The final step in the migration was to integrate our software into the existing routines that control the SS-411. Since our code was already neatly compartmentalized, this required very little change in our code, save some minor changes in memory allocation.

Algorithmic Optimization

Before we consider the details of the algorithm implementation, we must take a better look at the analytical model of sensor illumination. In previous sections, we have assumed that we have an analytical expression for $I_{mdl_1}(\tau)$, the single-slit illumination pattern. SAIL researchers have devised several image models of varying degrees of sophistication. Our most accurate model calculates diffraction through the slits from physical principals, and explicitly takes into account the nature of the sun's illumination and the size of the detector pixels. While useful in detailed simulation, it is impractically slow for on-line use. The second model we employed in the past is a simple Gaussian:

$$I_{mdl_1}(\tau) = e^{-\frac{\tau^2}{2\sigma^2}} \tag{17}$$

This has the advantage of a simple representation and parameterization (we need only find σ). While it provides reasonable good agreement with the peak shape, it is still not ideal for embedded applications since it requires the evaluation of the transcendental exponentiation function. Our final model candidate is a ratio of polynomials. Since we require a positive, symmetric function that drops quickly to zero, we include only even powers of the independent variable. Thus, our chosen image model becomes:

$$I_{mdl_1}(u) = \frac{c_1u^4 + c_2u^2 + c_3}{c_4u^4 + c_5u^2 + c_6} \tag{18}$$

Assuming that we cache u^2 and u^4, evaluating this ratio involves 6 multiplies, 4 adds, and one divide. With proper scaling this could possibly be performed in fixed-point arithmetic, but in our studies, only floating point operations were used. Coefficients were chosen to give a good fit to the peaks in sample images. This function and its derivatives effectively drop to zero outside of a narrow range around $u = 0$.

The main tasks that must be completed in each NLSQ iteration are the calculation of $\boldsymbol{S} = \boldsymbol{Q}^T\boldsymbol{Q}$, and $\boldsymbol{T} = \boldsymbol{Q}^T\boldsymbol{d}\beta$. Although $\boldsymbol{Q}$ is a 256×6 matrix, $\boldsymbol{S}$ is only 6×6 (for Type-0 images). If we can calculate $\boldsymbol{S}$ directly, rather than through the intermediate use of the $\boldsymbol{Q}$, matrix, we will save an enormous amount of space. The 36 elements, when stored as single-precision floating-point numbers would occupy 144 bytes of memory on the SS-411. This is an acceptable amount of storage given the constraints of this processor.

In order to calculate $\boldsymbol{S}$ directly we must consider its composition. Each element can be expressed as:

$$s_{jk} = \boldsymbol{q}_j \cdot \boldsymbol{q}_k \tag{19}$$

where the $\boldsymbol{q}_i$ quantities are the column vectors of $\boldsymbol{Q}$, and the LHS of the above equation is an inner product (i.e. a dot-product). Expanding the dot product explicitly, we get an expression in terms of scalar quantities:

$$s_{jk} = \sum_{n=1}^{256} \boldsymbol{q}_{\boldsymbol{j}_n} \cdot \boldsymbol{q}_{\boldsymbol{k}_n} = \sum_{n=1}^{256} q_j(\tau_n) \cdot q_k(\tau_n) \tag{20}$$

A similar expression applies to $\boldsymbol{T}$:

$$t_j = \sum_{n=1}^{256} q_{j_n} \cdot d\beta_n \tag{21}$$

Thus, we can incrementally calculate $\boldsymbol{S}$ and $\boldsymbol{T}$ by looping over the τ_n values. We do this efficiently by calculating:

$$dS_n = Q_n^T Q_n \tag{22}$$

where Q_n is the simply the n-th row of $\boldsymbol{Q}$. Thus, we calculate and store one row of $\boldsymbol{Q}$ at a time, and use this to incrementally calculate $\boldsymbol{S}$ and $\boldsymbol{T}$.

Furthermore, since the component functions (e.g. $\boldsymbol{A}, \boldsymbol{B}, \ldots$) that constitute the q_j are zero for most values of n, many of the floating-point operations required to calculate dS_n and the polynomial evaluations needed for q_{j_n} can be avoided if we check the input τ_n before evaluating I_{mdl} or its derivatives.

Floating Point Optimization

To get the very best real-time performance out of the sensor algorithms the C library floating point multiply routine was re-written in assembly language. A significant speed increase was realized (Table 3).

TABLE 3: Summary of Floating Point Optimization

C library FP multiply	270 clocks (nominal)
Hand-Optimized FP multiply	175 clocks (nominal)
Multiply Speed Increase	35%
Overall Algorithm Speed Increase	12%

These results were possible not because the original C libraries were poorly written, but because they were overly generic. All 8051 family processors execute common code but the timings for each opcode vary. Some of the speed increase comes from emphasizing those instructions that this processor executes particularly quickly. Further gains come from deviating from the strict IEEE-754 floating point standard. In addition to real numbers the standard supports various denormalized, infinity, and not-a-number conditions. The sensor algorithms do not require these, and with the special case code removed the real-time performance is improved.

Real-Time Performance

After careful optimization, the program size, RAM usage, and computational demands of these algorithms fit comfortably within capabilities of the SS-411. Timing measurements on the sensor indicated that the image segmentation takes less than 5 ms to complete. Each iteration of the NLSQ algorithm (including the linear solve and the variable update) takes 15.4 ms. Assuming that the user allows a maximum of 10 NLSQ iterations (a typical value), the sensor should be capable of taking full-accuracy measurements at about 5 Hz.

LABORATORY TESTING

Before delivering the improved software for the SS-411, extensive laboratory testing was required. These tests served two purposes. The sensors were *calibrated* to determine numerical constants used in the signal processing routines. The sensors were also *validated* to ensure that the software operated correctly. These tests were all conducted in the Space Avionics and Instrumentation Laboratory (SAIL) at Ryerson University.

The Ryerson SAIL Facility

The SAIL facility in the Department of Aerospace Engineering at Ryerson University, is equipped to simulate, prototype, and test spacecraft attitude sensors. SAIL research emphasizes the application of intelligent signal processing to improve the cost-specific performance of microsatellite sensors. To date, most of the SAIL research has concentrated on digital sun-sensors, but we are currently establishing capabilities to work with other devices (e.g. medium-accuracy star sensors, LIDAR, Earth sensors, etc.). Our experimental facilities leverage flexible, automated data collection to provide high-quality, accessible testing.

A schematic of the experimental setup for the SS-411 is shown in Figure 11. Sensor illumination is provided by a xenon arc-lamp, a common substitute for the sun. A condenser lens is used to collimate and direct the light towards the sensor. An adjustable iris restricts the apparent angular diameter of the light to that of the sun (i.e., 0.52°). It is difficult to accurately replicate solar geometry, wavefront properties, spectrum, and flux in the lab. This setup prioritizes geometry and wavefront, at the expense of flux. The sensor is able to compensate for this by increasing its integration time.

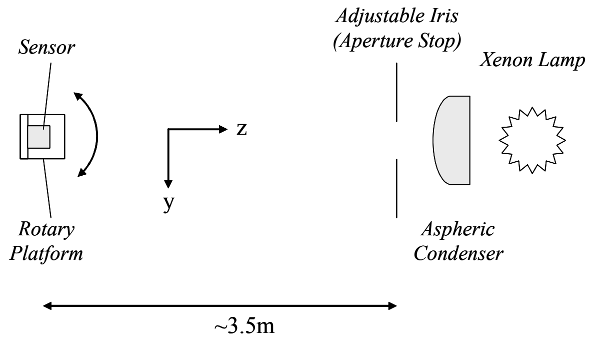

Figure 11: Diagram of experimental setup. The light source and sensor platform are mounted on an optics table.

The sensor is mounted to on a three-axis rotary test platform (Figure 12). The three motion stages are computer controlled and can be commanded in 0.001° increments (Absolute repeatability: 0.002°, 1-σ). Tests sequences are completely automated and can be controlled through the local network.

Sensor Calibration

Imperfections introduced during assembly, or by the laboratory setup must be measured as part of the calibration process. Several of the quantities introduced in the optics model (h, ρ_x, ρ_y, ψ, ϕ) have ranges associated with them. The actual values of these parameters must be measured if the on-board algorithms are to work correctly. These *sensor parameters* capture variations introduced by component and assembly tolerances and allow the calculation of $\tilde{s}_0$.

Furthermore, the alignment (rotation and translation) of the sensor with respect to the test platform must also be corrected during calibration. These *lab parameters* simplify test setup and reduce the need for high precision alignment, but complicate the calibration process. These

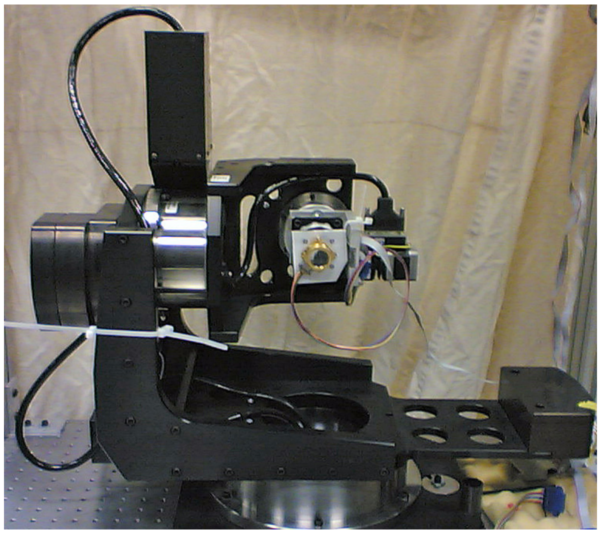

Figure 12: A sensor under evaluation on the SAIL rotary test platform

parameters are used to calculate the true incident sun vector, $\hat{s}_0$. To understand this latter class of parameters, we first considering the kinematics of the sensor platform. We calculate the incident sun vector using the following relation:

$$\vec{r}_{sun_S} = R_x(\eta_1)R_y(\eta_2)(K^{-1}\vec{r}_{1_L} + \vec{r}_{2_P}) \tag{23}$$

where $R_j(\theta)$ is a principal axis rotation about axis j, $\hat{r}_1$ is a vector from the centre of rotation of the test platform, to the sun lamp, $\hat{r}_2$ is the offset between the sensor origin and the centre of rotation, the η_i's capture tip-tilt mounting error and K is a transformation matrix derived from the kinematics of the sensor test platform. Normalizing this vector gives $\hat{s}_0$ with respect to the external sensor frame.

To calibrate a sensor, we first take a set of measurements (~400 points), at locations spread throughout the field of view. This data is used to solve for values of the calibration parameters that minimize the RMS estimation error over the test sequence. This procedure is performed offline. Solving for these quantities allows us to accurately calculate the incident sun vector $\hat{s}_0$ for any given set of joint angles. We do not need to precisely align the lamp beam with the centre of rotation, nor is the sensor mounting angle crucial — these are determined automatically during calibration. A summary of the quantities used during calibration is shown in Table 4.

Sensor Validation

After performing the sensor calibration, the residual error gives a measure of the quality of the parameter fit. Plotting the residual calibration error as a function of position in the field of view can give insight into the sensor behaviour. The results presented here are for a particular SS-411 sensor, but we feel that they are indicative of the design.

TABLE 4: Summary of Model Parameters

Parameter	Unit	Type	Typical Value
ϕ	rad	Sensor	0
ψ	rad	Sensor	$\pi/4$
ρ_x	m	Sensor	-5.7×10^{-3}
ρ_y	m	Sensor	-5.7×10^{-3}
h	m	Sensor	4.0×10^{-3}
$\vec{r}_1 = \begin{bmatrix} r_{1_x} & r_{1_y} & 1 \end{bmatrix}^T$	N/A	Lab	$\sim 1\times10^{-3}$
$\vec{r}_2 = \begin{bmatrix} r_{2_x} & r_{2_y} & r_{2_z} \end{bmatrix}^T$	N/A	Lab	$\sim 1\times10^{-2}$
η_1, η_2	rad	Lab	$< 5\times10^{-3}$
D_x	m	Fixed	7.5×10^{-4}
D_y	m	Fixed	2.5×10^{-4}
n_{glass}	N/A	Fixed	1.51
ΔX	m	Fixed	63.5×10^{-6}

Figure 13 shows the accuracy of the SS-411 sensor. The residual error, weighted appropriately by solid angle, was 0.12°. The blank area in the lower right represents a few data points where the sensor housing shadows one or more of the slits. These points were omitted from the data reduction.

Two important observations can be made from the data. First, the performance degrades at the outer edge of the field of view, but is otherwise fairly insensitive to distance from the boresight. Second, there are horizontal bands of noticeably poorer performance. Although the overall fit to the data is quite good, these bands suggest the presence of uncorrected systematic error. This strongly suggests a correlation between the y-axis component of the sun vector and the error in the attitude estimate. Probable causes of this error are discussed in the following section.

After performing the sensor calibration the on-board software must be updated with the appropriate sensor constants. Once these have been loaded, we perform a validation test sequence. This test is usually shorter (~100 points), and does not repeat the sample locations

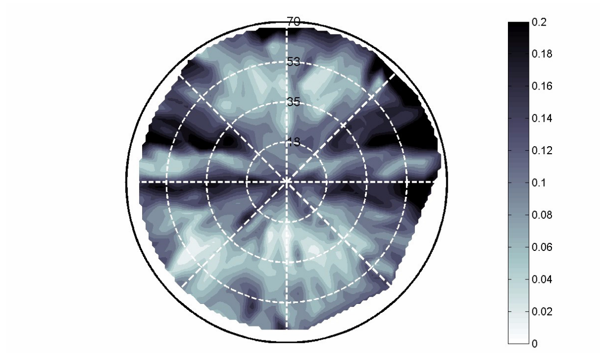

Figure 13: Sensor error throughout the field of view. Error bar is in degrees.

of the calibration sequence. During this test, the on-line estimates are collected. The mean, solid-angle weighted error for this test was 0.16°.

This error is noticeably larger than the residual error from the calibration sequence. The precise mechanism for this drop in performance is presently unclear but it does not appear to be a problem with the algorithm implementation. Off-line estimates at the validation points agree with on-line estimates to within less than 0.001°. Instead, we suspect deficiencies in the physical models of the optics and mounting geometry. If our models are not truly representative of the optical behaviour of the device, the residual errors from the calibration process will show evidence of structure.

Systematic Error in Peak Position Estimates

Horizontal bands of high error were apparent in the residual error plot after calibration (Figure 13). This suggests a correlation between y-component of the sun-vector, and the estimation error. Taking a closer look at the results from this axis reveals several clues to the origin of this error. The PPE error shows a clear trend when plotted against the mean position of the y-axis peaks (Figure 14a). The error seems linear on either end of the array, with a break in the middle. Furthermore, if we carefully measure the spacing between y-axis peaks with variant of the NLSQ algorithm, we notice an apparent increase in peak spacing near the centre of the array (Figure 14b).

Although the geometric sensor model treats the detector as a single monolithic array, the physical device is actually composed of two, 128-pixel sub-arrays. Close examination of the detectors reveal that not only is there a visible joint between segments, but that the pixels near the gap are also elongated (Figure 15). On a macro scale, a consistent pixel-pitch seems to be maintained, but it is clear that measurements near the seam will

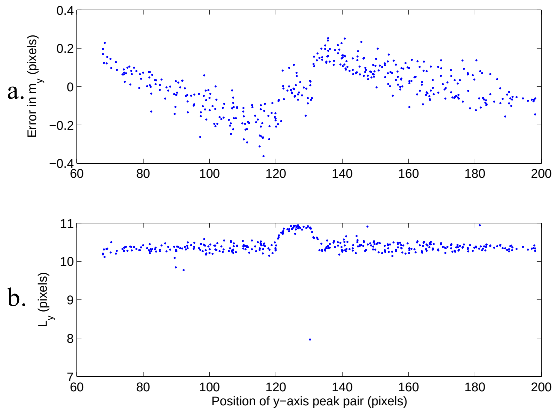

Figure 14: Systematic error in attitude estimates.

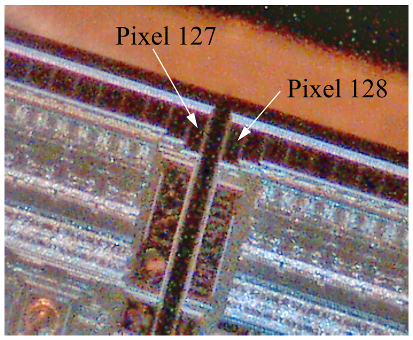

Figure 15: The sensor detector at 200x magnification. The 256 element detector is composed of two, 128-pixel segments.

experience some distortion. Given that the algorithms are trying to make measurements to a fraction of a pixel, it is unsurprising that this discovery will have an effect on the calibration. At time of writing, we have not fully explored the performance implications of this gap. It appears, however, that in trying to minimize the mean-squared error, the calibration process settled on non-physical parameter values that introduced the systematic error into the attitude estimates. In time, we expect that a suitable correction can be found.

CONCLUSIONS

When designing the test apparatus for the laboratory experiments it rapidly became apparent that high precision alignment of the motion platform, the sun lamp, and even the sensors themselves presented a significant challenge. We found that a more effective use of personnel and equipment was to perform a 'best-effort' alignment, followed by a comprehensive calibration. Off-line computation is cheap and fast, so calculating twelve parameters was no more difficult than five.

Explicitly modelling non-idealities such as sensor offset provided the means of removing the effects of these phenomena without having to eliminate the phenomena themselves. A reasonable degree of care during setup was still necessary to ensure that the calibration converged, but aligning to within a handful of mm or a few degrees was usually easy to do. This approach *does* require good physical insight into the sources of error, but simple approximations can go a long way.

Where calibration breaks down, is when the models fail to represent the underlying physics. This is illustrated in the difficulties encountered with the two-segment detector array. The fitting process selects the parameter values that minimize the RMS error, whether or not these are actuate measurements of real physical quantities. As a consequence, trends in the residual are often distorted and hard to decipher.

Further development efforts for the SS-411 will concentrate on software revisions and validation. The optical design of the SS-411 is sufficiently mature that further revisions are not pressing. Some unmodelled non-idealities, particularly the gap in the detector array, have noticeable effects on performance. We are currently investigating the best way to compensate for this effect. In the mid-term, once some of these sensors have been launched, flight data can be used to provide additional validation and an opportunity to study operational performance in the presence of effects such as albedo. In the long-term, the sensor performance may be reaching the point of diminishing returns. The utility of increased precision is hampered by the fundamental limitation of sun sensors — that they only provide two components of attitude knowledge. Magnetometers are not accurate enough to effectively complement a high accuracy sun-sensor, and star-trackers provide three-axis measurements on their own. Simulations of the SS-411 suggest that achieving a mean error of 0.05° is possible. This likely represents a practical limit to the usefulness of these devices and is the current goal of our research.

ACKNOWLEDGMENTS

Funding for this work provided in part by the Canadian Space Agency and the Ontario Centres of Excellence

TABLE 5: Nomenclature

a_1, a_2, b_1, b_2	modeled image component amplitude
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$	modeled image components
D_x, D_y	coordinate of slits along x/y-axis
$\boldsymbol{d\lambda}$	update to NLSQ parameters
$\boldsymbol{E}, \boldsymbol{F}, \boldsymbol{G}, \boldsymbol{H}$	slope of modeled image components
h	thickness of optics, m
I	sensor image
K	rotation matrix from test-platform kinematics
L	half spacing between peak pairs, on detector, m
m	peak location, pixels
n	pixel index
N	number of pixels
n_{glass}	index of refraction for glass
p	set of peak locations
$\boldsymbol{Q}$	non-linear, least squares covariance matrix
$R_i(\alpha)$	principal axis rotation about axis-i
$\hat{s}_0$	freespace unit vector from the sun
$\tilde{s}_0$	estimated freespace vector
$\vec{r}_1$	vector from platform centre-of-rotation to lamp
$\vec{r}_2$	vector from sensor origin, to test platform origin
β	error between NLSQ image model and real image
ΔX	pixel spacing, m
η_1, η_2	tip/tilt parameters for mounting model, radians
ρ_x, ρ_y	location of pixel-0 in image plane
τ	displacement along detector array, from pixel-0, m
τ_n	displacement of pixel n
ϕ	angle between aperture slits and sensor housing
ψ	angle between x-axis and sensor detector array

REFERENCES

1. Enright, J., Li, C., Yam, A., “Modelling and Testing of Two-Dimensional Sun-Sensors”, *Proc. of the IEEE Aerospace Conference*, March 3-10, 2007, Big Sky, MT

2. Enright, J., Godard, “Super-Resolution Techniques for Sun-Sensor Processing”, accepted to *IEEE Transactions on Aerospace and Electronic Systems*.

3. Liebe, C.; Mobasser, S.; Bae, Y.; Wrigley, C.; Schroeder, J.; Howard, A.; “Micro Sun Sensor”, *Proc. of IEEE Aerospace Conference*, Vol. 5, pg 2263-2273, March 2002.

4. de Boom, C., van der Heiden, N., “A Novel Digital Sun Sensor: Development and Qualification for Flight”, *54th International Astronautical Congress*, Sept.-Oct. 2003, IAC-03-A.P.20, Bremen, Germany

5. Ninomiya, K., Ogawara, Y., Tsuno, K., Akabane, S.; “High Accuracy Sun Sensor Using CCDs”; *AIAA Guidance, Navigation and Control Conference*, Minneapolis, MN, Aug 15-17, 1988, p. 1061-1070

6. Data sheet available on the Silicon Labs website: *http://www.silabs.com/public/documents/tpub_doc/dsheet/Microcontrollers/Small_Form_Factor/en/C8051F41x.pdf*

7. Bates, D. M. and Watts, D. G.; *Nonlinear Regression and Its Applications.* New York: Wiley, 1988.