Modeling and Adjustment of THEMIS IR
Line Scanner Camera Image Measurements

by

Brent Archinal
USGS Astrogeology Team
2255 N. Gemini Drive
Flagstaff, AZ 86001

barchinal@usgs.gov

As of 2004 December 9

Version 1.0

Table of Contents

1. Introduction

1.1 General

The overall problem we're solving is that we want to be able to set up the relationships between the coordinates of arbitrary physical points in space (e.g. ground points) and their coordinates on line scanner (or "pushbroom") camera images. We then want to do a least squares solution in order to come up with consistent camera orientation and position information that represents these relationships accurately.

For now, supported by funding from the NASA Critical Data Products initiative (for 2003 September to 2005 August), we will concentrate on handling the THEMIS IR camera system (Christensen et al., 2003). Although it is in fact a multispectral line scanner camera system, we will (generally) treat it for now as a single line scanner camera system. (At some future point the multiple spectral images sensors could allow this system to be treated as a multi-line scanner camera system.) As much as possible, we will also keep our options open for processing of other planetary line scanner camera systems. In particular, in the near term of the next 1-2 years, we plan to make modifications to allow for processing of images from first the Mars Express HRSC camera system (a 9 line scanner camera system, with 3 sets of 3 lines for color) (Neukum et al., 2004) and secondly the Mars Reconnaissance Orbiter (MRO) HiRISE camera system (McEwen et al., 2002; Kirk, 2004 – see Appendix II). In the longer term we may seek funding to allow for processing of Mars Global Surveyor (MGS) Mars Orbiter Camera (MOC) wide angle (WA) and narrow angle (NA) cameras (Malin et al., 1992; Malin and Edgett, 2001), as well as the Cassini UVIS camera. We may also consider allowing for processing of images from the multispectral Cassini VIMS pixel scanner ("whiskbroom") system.

The basic processing steps are essentially the same as those required for a framing camera system, e.g. as already implemented in the RAND-USGS Planetary Geodesy (RUPG) software system (Colvin, 1992; Archinal et al, 2002-2004). The essential difference is that it becomes necessary to solve for the position and (external) orientation of the camera for every line (or pixel for a pixel scanner) rather than just once per frame. Using standard techniques such a problem is substantially underdetermined, particularly using any reasonable number of tie point measurements per image. The straightforward solution – in theory but not necessarily in practice – is to add parameters that model the camera position and orientation changes as the image is collected, as offsets from one or more sets of discrete values (e.g. the one set a framing camera would have).

A search of the literature shows a nearly overwhelming number of ways in which to model the motions taking place during image collection. A difficult part of the implementation here has indeed been to select one of the methods used.

Methods developed previously have had varying dependencies on the specific camera system, e.g. whether airborne or satellite, number of lines, availability of auxiliary information, accuracy and time density of orbit and orientation information, size of the image, etc.

Common methods previously used seem to be categorized into: a) fitting smooth curves of position and orientation changes between discrete lines (e.g. breaking up the image into multiple sections, and for each section relying on measured satellite position via GPS and orientation via some inertial system) (e.g. Poli et al., 2004); b) fitting simple polynomial offsets for position and orientation (e.g. Kratky, 1989; Fritsch and Stallmann, 2002); c) fitting Lagrangian interpolation polynomial offsets for position and orientation (which seems to be preferred over spline interpolation methods); and d) fitting offsets for position by one of the other methods and for orientation with Fourier series offsets (partially to allow for recovery of high frequency jitter, e.g. with Oberst's (2002) HWBUNDLE *lite* program). Reviews of possible methods are given for example by Gruen and Zhang (2003); Poli et al. (2004), and Toutin (2004).

The orbits are also modeled either as simply a function of Cartesian coordinates, or for longer arcs or to better represent the orbit parameters uncertainties or to allow for the use in more complicated orbit solutions, as a function of Keplerian elements.

Finally, additional parameters are required in order to represent the geometry of each line of a line scanner camera relative the camera boresight, and therefore in effect the relative line scanner positions.

For the purpose of implementing a line scanner camera adjustment procedure for THEMIS IR, we will use second order polynomials to represent camera (spacecraft) Cartesian coordinates and orientation. Initially, we will assume that camera calibration information is known (e.g. as already used in ISIS). At a later date, we could extend our model to allow for the adjustment of the line scanner position relative to the boresight, as well as adjustment of the absolute start time of each image or image portion (if an image has missing lines), and other calibration parameters as necessary. This model could also be extended to allow for the use of other models of interpolation of position and orientation between lines, as the need arises for other missions. See Section 6 below for a discussion of this and other possible future extensions.

For the purpose of software implementation, it has been assumed that the RUPG *randlsq* program would be modified. Other options exist, such as rewriting that program, writing something from scratch, or waiting until the adjustment program planned for DISR and MER processing is complete – and then modifying it. However, the *randlsq* modification option currently appears to be the most practical and time- and cost-effective, for reasons discussed further in Section 4.

In the rest of this paper, material on the specific new parameters and how relevant partial derivatives should be computed is presented in Section 2. In section 3 I provide briefly – more as a review reference then anything else - the equations for how these observation equations and their partials can be used in a least squares adjustment in order to recover estimates of improved parameter values and their variances and covariances. Section 4 considers what specific changes will be needed to the *randlsq* software and suggests how these could be implemented. Section 5 briefly discusses the testing we plan to do, and lists a few items for planned near-term implementation. Section 6 discusses items that could be or will be (e.g. handling images from other missions) implemented in the longer term.

This write-up concludes with a list of references, not all cited here, but including many regarding the RUPG software, and line scanner camera data handling. The usefulness of these references is annotated in the list (in brackets). Appendix I reproduces the first part of Colvin's (1992) write-up concerning the theory used in what has become the *randlsq* program. This is important not only since it explains to a reasonable extent the implementation/code in *randlsq*, but since Section 2 here is based on the same formulation. Appendix II reproduces a recent e-mail from R. Kirk which describes the HiRISE camera model and therefore includes examples of the type of camera calibration parameters needed for a (multi) line-scanner camera system.

For information on ISIS, see Eliason (1997); Gaddis et al. (1997); Torson and Becker (1997) and http://isis.astrogeology.usgs.gov/.

1.2 Conventions

Vectors and matrices are shown as a single bold variable.

Note that (so far) in the equations presented following, there is no distinction between image measurements of the same point on different images, or on different images from different cameras. Bookkeeping of this information will obviously need to be done (i.e. extended as needed) in the *randlsq* software.

Also note that in *randlsq*, x increases with sample values (across the image) and y increases with line values (along the image), as "sample" and "line" is defined in the ISIS *qmatch* software (e-mail from B. Archinal to R. Sucharski of 2003 January 27).

2. Observations Equations and Their Partials

The following discussion is based on that given by Colvin (1992) which is the basis for the current *randlsq* program. For reference, Colvin's discussion is reproduced below in Appendix I. It is based on the assumption that one starts with the ground point coordinates, and then the transformation through successive coordinate systems is derived, back to the camera image plane (x, y) measurements. Note that this is different from the more common photogrammetric solution presentation, where the overall transformation – the collinearity equations – from image to ground coordinates is presented first, and then the component transformations derived in detail. For an example of this sort of presentation, see e.g. McGlone (2004). Needed partial derivatives are derived along the way, with the derivatives of the image plane measurements with respect to the parameters then computed as needed, e.g. using the chain rule.

So rather then derive everything anew, what is given here is an extension of the presentation by Colvin. In section 2.1 the additional parameters and model needed to model a line scanner camera are presented. In section 2.2 the needed additional partial derivatives are derived. These collectively can then be used to modify the *randlsq* program in order to adjust image measures (the basic observations) of line scanner camera images.

2.1 Line Scanner Camera Specific Modeling

Let's assume the inertial position of the spacecraft in the body (Mars) centered system is given by X_S, Y_S, and Z_S. The boresight of the camera (for a given line) is pointed toward the (quasi-inertial) ICRF (i.e. "J2000.0") right ascension and declination $\hat{\alpha}$, $\hat{\delta}$, and the line is rotated from north by the twist angle κ. This is all at time t.

We will assume that (up to) second order polynomial variations are possible in the spacecraft position and pointing. This gives us (McGlone, pp. 290-291, equations 3.138 and 3.139):

$$\begin{bmatrix} X_s \\ Y_s \\ Z_s \end{bmatrix} = \begin{bmatrix} X_0 + X_1(t-t_R) + X_2(t-t_R)^2 \\ Y_0 + Y_1(t-t_R) + Y_2(t-t_R)^2 \\ Z_0 + Z_1(t-t_R) + Z_2(t-t_R)^2 \end{bmatrix} \qquad (1)$$

$$\begin{bmatrix} \hat{\alpha} \\ \hat{\delta} \\ \kappa \end{bmatrix} = \begin{bmatrix} \hat{\alpha}_0 + \hat{\alpha}_1(t-t_R) + \hat{\alpha}_2(t-t_R)^2 \\ \hat{\delta}_0 + \hat{\delta}_1(t-t_R) + \hat{\delta}_2(t-t_R)^2 \\ \kappa_0 + \kappa_1(t-t_R) + \kappa_2(t-t_R)^2 \end{bmatrix} \qquad (2)$$

Where X_0, Y_0, and Z_0 is the position of the spacecraft at some reference time t_R, which we will assume here is the time of the central line of a particular image. Similarly, the camera orientation at that moment is given by the J2000.0 right ascension and declination $\hat{\alpha}_0$, $\hat{\delta}_0$, and twist angle κ_0.

To allow for unknown variations in the spacecraft position and orientation while the image is being obtained, we will solve for some or all of additional unknowns X_1, Y_1, Z_1, X_2, Y_2, Z_2, $\hat{\alpha}_1$, $\hat{\delta}_1$, κ_1, $\hat{\alpha}_2$, $\hat{\delta}_2$, and κ_2.

Some experimentation will be needed in order to determine what the best parameter set is. There are several possible cases:

1. Currently, for framing cameras, we solve only for $\hat{\alpha}_0$, $\hat{\delta}_0$, and κ_0 for each image. Solutions with only these parameters can/should be done to provide checks that the algorithm and software are working, and for comparison with solutions with additional parameters.
2. Solving for the satellite reference position X_0, Y_0, and Z_0 would be the next step, although when these parameters are solved for it is likely they will need to be constrained (weighted) at their likely level of accuracy. See Section 2.2.4 below for additional comments on this.
3. Solving for rates of change in the spacecraft position and (camera) orientation would likely be next, i.e. for the coefficients X_1, Y_1, Z_1, $\hat{\alpha}_1$, $\hat{\delta}_1$, and κ_1.
4. For particularly long images (many lines) solving for acceleration of the spacecraft position and camera orientation may possibly be necessary, i.e. for the coefficients X_2, Y_2, Z_2, $\hat{\alpha}_2$, $\hat{\delta}_2$, and κ_2.

Ultimately, only testing will show whether this overall polynomial model is satisfactory. It may be necessary to substitute some other types of models for equations 1 and 2, e.g. fitting some other types of polynomials, Fourier series (Oberst, 2002), or perhaps even parameters that correspond to changes in Keplerian orbit parameters. Such changes may also be necessary for other spacecraft/camera combinations, e.g. with known "jitter" problems (e.g. MGS/MOC) or higher resolutions (e.g. MRO/HiRISE).

For multiple line scanner cameras, we need only impose two additional requirements to our model. First, that the time *t* of any lines is referenced to a reference line in <u>one</u> of the line scanner images. Secondly, orientation need only be solved for for one of the images being collected by the camera, with the orientation of the other images fixed by a set offset angle (or set of rotations). (Although it may also be necessary to solve for this angle or angles for calibration purposes.)

It may also be useful to add parameters for additional camera calibration, e.g. the focal length *f*, of the center of the line, a scale factor along the line reflecting any change in focal length along the line or tilt of the linear array relative to the focal plane, the offset of the center of the line from the camera boresight, i.e. x_0 and y_0, and possibly other camera constants (e.g. for distortion). As an example and for future use here, this type of parameterization is discussed by R. Kirk for the HiRISE camera (e-mail to HiGeom group mailing list of 2004 November 22 – see Appendix II).

2.2 Partials for New Parameters

As mentioned above, the only new parameters being solved for in comparison to a solution using measurements from a framing camera are (for each image): X_0, Y_0, Z_0, X_1, Y_1, Z_1, X_2, Y_2, Z_2, $\hat{\alpha}_1$, $\hat{\delta}_1$, κ_1, $\hat{\alpha}_2$, $\hat{\delta}_2$, and κ_2.

We therefore need to modify the current *randlsq* algorithm/program to compute partial derivatives of the observation equations for these parameters (in addition to the partials being computed for existing parameters), and to use these partials to set up the solution for the selected parameters.

2.2.1 Orientation Partials

We already have (from (Colvin, 1992, equation 9 and following) the definition of the camera orientation matrix **C** and its partial derivatives with respect to the three camera angles, i.e. **C**, $\frac{\partial \mathbf{C}}{\partial \hat{\alpha}}$, $\frac{\partial \mathbf{C}}{\partial \hat{\delta}}$, and $\frac{\partial \mathbf{C}}{\partial \kappa}$.

Using this as a starting point, we need to compute:

$$\frac{\partial \mathbf{C}}{\partial \hat{\alpha}_0}, \frac{\partial \mathbf{C}}{\partial \hat{\delta}_0}, \frac{\partial \mathbf{C}}{\partial \kappa_0}, \frac{\partial \mathbf{C}}{\partial \hat{\alpha}_1}, \frac{\partial \mathbf{C}}{\partial \hat{\delta}_1}, \frac{\partial \mathbf{C}}{\partial \kappa_1}, \frac{\partial \mathbf{C}}{\partial \hat{\alpha}_2}, \frac{\partial \mathbf{C}}{\partial \hat{\delta}_2}, \text{ and } \frac{\partial \mathbf{C}}{\partial \kappa_2}.$$

We also know (from (2) above) that:

$$\begin{bmatrix} \frac{\partial \hat{\alpha}}{\partial \hat{\alpha}_0} \\ \frac{\partial \hat{\delta}}{\partial \hat{\delta}_0} \\ \frac{\partial \kappa}{\partial \kappa_0} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad (3)$$

$$\begin{bmatrix} \frac{\partial \hat{\alpha}}{\partial \hat{\alpha}_1} \\ \frac{\partial \hat{\delta}}{\partial \hat{\delta}_1} \\ \frac{\partial \kappa}{\partial \kappa_1} \end{bmatrix} = \begin{bmatrix} t - t_R \\ t - t_R \\ t - t_R \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} \frac{\partial \hat{\alpha}}{\partial \hat{\alpha}_2} \\ \frac{\partial \hat{\delta}}{\partial \hat{\delta}_2} \\ \frac{\partial \kappa}{\partial \kappa_2} \end{bmatrix} = \begin{bmatrix} (t - t_R)^2 \\ (t - t_R)^2 \\ (t - t_R)^2 \end{bmatrix} \tag{5}$$

So from Colvin (1992) equation 9 we have:

$$\mathbf{C} = \begin{bmatrix} -\sin\hat{\alpha}\cos\kappa - \cos\hat{\alpha}\sin\hat{\delta}\sin\kappa & \sin\hat{\alpha}\cos\kappa - \sin\hat{\alpha}\sin\hat{\delta}\sin\kappa & \cos\hat{\delta}\sin\kappa \\ \sin\hat{\alpha}\sin\kappa - \cos\hat{\alpha}\sin\hat{\delta}\cos\kappa & -\cos\hat{\alpha}\sin\kappa - \sin\hat{\alpha}\sin\hat{\delta}\cos\kappa & \cos\hat{\delta}\cos\kappa \\ \cos\hat{\alpha}\cos\hat{\delta} & \sin\hat{\alpha}\cos\hat{\delta} & \sin\hat{\delta} \end{bmatrix}$$

and the partials with respect to each angle:

$$\frac{\partial \mathbf{C}}{\partial \hat{\alpha}} = \begin{bmatrix} -\cos\hat{\alpha}\cos\kappa + \sin\hat{\alpha}\sin\hat{\delta}\sin\kappa & -\sin\hat{\alpha}\cos\kappa - \cos\hat{\alpha}\sin\hat{\delta}\sin\kappa & 0 \\ \cos\hat{\alpha}\sin\kappa + \sin\hat{\alpha}\sin\hat{\delta}\cos\kappa & \sin\hat{\alpha}\sin\kappa - \cos\hat{\alpha}\sin\hat{\delta}\cos\kappa & 0 \\ -\sin\hat{\alpha}\cos\hat{\delta} & \cos\hat{\alpha}\cos\hat{\delta} & 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{C}}{\partial \hat{\delta}} = \begin{bmatrix} -\cos\hat{\alpha}\cos\hat{\delta}\sin\kappa & -\sin\hat{\alpha}\cos\hat{\delta}\sin\kappa & -\sin\hat{\delta}\sin\kappa \\ -\cos\hat{\alpha}\cos\hat{\delta}\cos\kappa & -\sin\hat{\alpha}\cos\hat{\delta}\cos\kappa & -\sin\hat{\delta}\cos\kappa \\ -\cos\hat{\alpha}\sin\hat{\delta} & -\sin\hat{\alpha}\sin\hat{\delta} & \cos\hat{\delta} \end{bmatrix}$$

$$\frac{\partial \mathbf{C}}{\partial \kappa} = \begin{bmatrix} \sin\hat{\alpha}\sin\kappa - \cos\hat{\alpha}\sin\hat{\delta}\cos\kappa & -\cos\hat{\alpha}\sin\kappa - \sin\hat{\alpha}\sin\hat{\delta}\cos\kappa & \cos\hat{\delta}\cos\kappa \\ \sin\hat{\alpha}\cos\kappa + \cos\hat{\alpha}\sin\hat{\delta}\sin\kappa & -\cos\hat{\alpha}\cos\kappa + \sin\hat{\alpha}\sin\hat{\delta}\sin\kappa & -\cos\hat{\delta}\sin\kappa \\ 0 & 0 & 0 \end{bmatrix}$$

As noted earlier, the $\hat{\alpha}_0$, $\hat{\delta}_0$, and κ_0 parameters actually correspond exactly to the $\hat{\alpha}$, $\hat{\delta}$, and κ parameters of the framing camera case, and the partials with respect to them for $\hat{\alpha}_0$, $\hat{\delta}_0$, and κ_0 are (from equation 3) just 1. So the partials of

C with respect to $\hat{\alpha}_0$, $\hat{\delta}_0$, and κ_0 are simply the above partials with the angle of interest substituted:

$$\frac{\partial \mathbf{C}}{\partial \hat{\alpha}_0} = \begin{bmatrix} -\cos\hat{\alpha}_0\cos\kappa + \sin\hat{\alpha}_0\sin\hat{\delta}\sin\kappa & -\sin\hat{\alpha}_0\cos\kappa - \cos\hat{\alpha}_0\sin\hat{\delta}\sin\kappa & 0 \\ \cos\hat{\alpha}_0\sin\kappa + \sin\hat{\alpha}_0\sin\hat{\delta}\cos\kappa & \sin\hat{\alpha}_0\sin\kappa - \cos\hat{\alpha}_0\sin\hat{\delta}\cos\kappa & 0 \\ -\sin\hat{\alpha}_0\cos\hat{\delta} & \cos\hat{\alpha}_0\cos\hat{\delta} & 0 \end{bmatrix} \quad (6)$$

$$\frac{\partial \mathbf{C}}{\partial \hat{\delta}_0} = \begin{bmatrix} -\cos\hat{\alpha}\cos\hat{\delta}_0\sin\kappa & -\sin\hat{\alpha}\cos\hat{\delta}_0\sin\kappa & -\sin\hat{\delta}_0\sin\kappa \\ -\cos\hat{\alpha}\cos\hat{\delta}_0\cos\kappa & -\sin\hat{\alpha}\cos\hat{\delta}_0\cos\kappa & -\sin\hat{\delta}_0\cos\kappa \\ -\cos\hat{\alpha}\sin\hat{\delta}_0 & -\sin\hat{\alpha}\sin\hat{\delta}_0 & \cos\hat{\delta}_0 \end{bmatrix} \quad (7)$$

$$\frac{\partial \mathbf{C}}{\partial \kappa_0} = \begin{bmatrix} \sin\hat{\alpha}\sin\kappa_0 - \cos\hat{\alpha}\sin\hat{\delta}\cos\kappa_0 & -\cos\hat{\alpha}\sin\kappa_0 - \sin\hat{\alpha}\sin\hat{\delta}\cos\kappa_0 & \cos\hat{\delta}\cos\kappa_0 \\ \sin\hat{\alpha}\cos\kappa_0 + \cos\hat{\alpha}\sin\hat{\delta}\sin\kappa_0 & -\cos\hat{\alpha}\cos\kappa_0 + \sin\hat{\alpha}\sin\hat{\delta}\sin\kappa_0 & -\cos\hat{\delta}\sin\kappa_0 \\ 0 & 0 & 0 \end{bmatrix} \quad (8)$$

The partials of $\hat{\alpha}$, $\hat{\delta}$, and κ with respect to $\hat{\alpha}_1$, $\hat{\delta}_1$, and κ_1 are (from equation 4) just ($t-t_R$), so the partials of C with respect to $\hat{\alpha}_1$, $\hat{\delta}_1$, and κ_1 become:

$$\frac{\partial \mathbf{C}}{\partial \hat{\alpha}_1} = \begin{bmatrix} (-\cos\hat{\alpha}_1\cos\kappa + \sin\hat{\alpha}_1\sin\hat{\delta}\sin\kappa)*(t-t_R) & (-\sin\hat{\alpha}_1\cos\kappa - \cos\hat{\alpha}_1\sin\hat{\delta}\sin\kappa)*(t-t_R) & 0 \\ (\cos\hat{\alpha}_1\sin\kappa + \sin\hat{\alpha}_1\sin\hat{\delta}\cos\kappa)*(t-t_R) & (\sin\hat{\alpha}_1\sin\kappa - \cos\hat{\alpha}_1\sin\hat{\delta}\cos\kappa)*(t-t_R) & 0 \\ (-\sin\hat{\alpha}_1\cos\hat{\delta})*(t-t_R) & (\cos\hat{\alpha}_1\cos\hat{\delta})*(t-t_R) & 0 \end{bmatrix} \quad (9)$$

$$\frac{\partial \mathbf{C}}{\partial \hat{\delta}_1} = \begin{bmatrix} (-\cos\hat{\alpha}\cos\hat{\delta}_1\sin\kappa)*(t-t_R) & (-\sin\hat{\alpha}\cos\hat{\delta}_1\sin\kappa)*(t-t_R) & (-\sin\hat{\delta}_1\sin\kappa)*(t-t_R) \\ (-\cos\hat{\alpha}\cos\hat{\delta}_1\cos\kappa)*(t-t_R) & (-\sin\hat{\alpha}\cos\hat{\delta}_1\cos\kappa)*(t-t_R) & (-\sin\hat{\delta}_1\cos\kappa)*(t-t_R) \\ (-\cos\hat{\alpha}\sin\hat{\delta}_1)*(t-t_R) & (-\sin\hat{\alpha}\sin\hat{\delta}_1)*(t-t_R) & \cos\hat{\delta}_1*(t-t_R) \end{bmatrix} \quad (10)$$

$$\frac{\partial \mathbf{C}}{\partial \kappa_1} = \begin{bmatrix} (\sin\hat{\alpha}\sin\kappa_1 - \cos\hat{\alpha}\sin\hat{\delta}\cos\kappa_1)*(t-t_R) & (-\cos\hat{\alpha}\sin\kappa_1 - \sin\hat{\alpha}\sin\hat{\delta}\cos\kappa_1)*(t-t_R) & \cos\hat{\delta}\cos\kappa_1*(t-t_R) \\ (\sin\hat{\alpha}\cos\kappa_1 + \cos\hat{\alpha}\sin\hat{\delta}\sin\kappa_1)*(t-t_R) & (-\cos\hat{\alpha}\cos\kappa_1 + \sin\hat{\alpha}\sin\hat{\delta}\sin\kappa_1)*(t-t_R) & -\cos\hat{\delta}\sin\kappa_1*(t-t_R) \\ 0 & 0 & 0 \end{bmatrix} \quad (11)$$

The partials of $\hat{\alpha}$, $\hat{\delta}$, and κ with respect to $\hat{\alpha}_2$, $\hat{\delta}_2$, and κ_2 are (from equation 4) just $(t-t_R)^2$, so the partials of C with respect to $\hat{\alpha}_2$, $\hat{\delta}_2$, and κ_2 become:

$$\frac{\partial \mathbf{C}}{\partial \alpha_2} = \begin{bmatrix} (-\cos\hat{\alpha}_2 \cos\kappa + \sin\hat{\alpha}_2 \sin\hat{\delta} \sin\kappa) * (t-t_R)^2 & (-\sin\hat{\alpha}_2 \cos\kappa - \cos\hat{\alpha}_2 \sin\hat{\delta} \sin\kappa) * (t-t_R)^2 & 0 \\ (\cos\hat{\alpha}_2 \sin\kappa + \sin\hat{\alpha}_2 \sin\hat{\delta} \cos\kappa) * (t-t_R)^2 & (\sin\hat{\alpha}_2 \sin\kappa - \cos\hat{\alpha}_2 \sin\hat{\delta} \cos\kappa) * (t-t_R)^2 & 0 \\ (-\sin\hat{\alpha}_2 \cos\hat{\delta}) * (t-t_R)^2 & (\cos\hat{\alpha}_2 \cos\hat{\delta}) * (t-t_R)^2 & 0 \end{bmatrix} \quad (12)$$

$$\frac{\partial \mathbf{C}}{\partial \hat{\delta}_2} = \begin{bmatrix} (-\cos\hat{\alpha} \cos\hat{\delta}_2 \sin\kappa) * (t-t_R)^2 & (-\sin\hat{\alpha} \cos\hat{\delta}_2 \sin\kappa) * (t-t_R)^2 & (-\sin\hat{\delta}_2 \sin\kappa) * (t-t_R)^2 \\ (-\cos\hat{\alpha} \cos\hat{\delta}_2 \cos\kappa) * (t-t_R)^2 & (-\sin\hat{\alpha} \cos\hat{\delta}_2 \cos\kappa) * (t-t_R)^2 & (-\sin\hat{\delta}_2 \cos\kappa) * (t-t_R)^2 \\ (-\cos\hat{\alpha} \sin\hat{\delta}_2) * (t-t_R)^2 & (-\sin\hat{\alpha} \sin\hat{\delta}_2) * (t-t_R)^2 & \cos\hat{\delta}_2 * (t-t_R)^2 \end{bmatrix} \quad (13)$$

$$\frac{\partial \mathbf{C}}{\partial \kappa_2} = \begin{bmatrix} (\sin\hat{\alpha} \sin\kappa_2 - \cos\hat{\alpha} \sin\hat{\delta} \cos\kappa_2) * (t-t_R)^2 & (-\cos\hat{\alpha} \sin\kappa_2 - \sin\hat{\alpha} \sin\hat{\delta} \cos\kappa_2) * (t-t_R)^2 & \cos\hat{\delta} \cos\kappa_2 * (t-t_R)^2 \\ (\sin\hat{\alpha} \cos\kappa_2 + \cos\hat{\alpha} \sin\hat{\delta} \sin\kappa_2) * (t-t_R)^2 & (-\cos\hat{\alpha} \cos\kappa_2 + \sin\hat{\alpha} \sin\hat{\delta} \sin\kappa_2) * (t-t_R)^2 & -\cos\hat{\delta} \sin\kappa_2 * (t-t_R)^2 \\ 0 & 0 & 0 \end{bmatrix} \quad (14)$$

2.2.2 Spatial Partials

Next we need to calculate the partials relative to spacecraft position, i.e. relative to X_0, Y_0, Z_0, X_1, Y_1, Z_1, X_2, Y_2, Z_2.

We have (from equation 1) the following:

$$\begin{bmatrix} \frac{\partial X_s}{\partial X_0} \\ \frac{\partial Y_s}{\partial Y_0} \\ \frac{\partial Z_s}{\partial Z_0} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (15)$$

$$\begin{bmatrix} \frac{\partial X_s}{\partial X_1} \\ \frac{\partial Y_s}{\partial Y_1} \\ \frac{\partial Z_s}{\partial Z_1} \end{bmatrix} = \begin{bmatrix} t - t_R \\ t - t_R \\ t - t_R \end{bmatrix} \quad (16)$$

$$\begin{bmatrix} \frac{\partial X_s}{\partial X_2} \\ \frac{\partial Y_s}{\partial Y_2} \\ \frac{\partial Z_s}{\partial Z_2} \end{bmatrix} = \begin{bmatrix} (t - t_R)^2 \\ (t - t_R)^2 \\ (t - t_R)^2 \end{bmatrix} \quad (17)$$

Colvin (1992, equation 8) defines the spacecraft position vector, with origin at the body center and in the J2000.0 coordinate system, as **s**. Therefore,

$$\mathbf{s} = \begin{bmatrix} X_S \\ Y_S \\ Z_s \end{bmatrix}$$

However, we are now solving for spacecraft position (and velocity and acceleration) so now must consider the partials of **s** with respect to the various parameters, i.e. in Colvin's terminology

$$\frac{\partial \mathbf{s}}{\partial P}, P \in \{\mathrm{X}_0, \mathrm{Y}_0, \mathrm{Z}_0, \mathrm{X}_1, \mathrm{Y}_1, \mathrm{Z}_1, \mathrm{X}_2, \mathrm{Y}_2, \mathrm{Z}_2\}$$

So if the range vector (the vector subtended by the spacecraft and a control point) is given by (Colvin, equation 8)

$$\mathbf{s}^{\mathrm{v}} = \hat{\mathbf{s}} - \mathbf{s}$$

then the partials of this with respect to the various parameters is now given by

$$\frac{\partial \mathbf{s}^v}{\partial P} = \frac{\partial \hat{\mathbf{s}}}{\partial P_1} - \frac{\partial \mathbf{s}}{\partial P_2}, P_1 \in \{\varphi, \lambda, R, \alpha_0, \delta_0, \dot{w}\}, P_2 \in \{\mathrm{X}_0, \mathrm{Y}_0, \mathrm{Z}_0, \mathrm{X}_1, \mathrm{Y}_1, \mathrm{Z}_1, \mathrm{X}_2, \mathrm{Y}_2, \mathrm{Z}_2\}$$

2.2.3 Partials of the observations with respect to the parameters

The range vector in camera coordinates is (Colvin's equation 10)

$$\gamma = \mathbf{C}\mathbf{s}^{\mathbf{v}}$$

and the partials of γ now become

$$\frac{\partial \gamma}{\partial P_1} = \mathbf{C}\frac{\partial \hat{\mathbf{s}}}{\partial P_1}, P_1 \in \{\varphi, \lambda, R, \alpha_0, \delta_0, \dot{w}\}$$
$$\frac{\partial \gamma}{\partial P_2} = -\mathbf{C}\frac{\partial \mathbf{s}}{\partial P_2}, P_2 \in \{\mathrm{X}_0, \mathrm{Y}_0, \mathrm{Z}_0, \mathrm{X}_1, \mathrm{Y}_1, \mathrm{Z}_1, \mathrm{X}_2, \mathrm{Y}_2, \mathrm{Z}_2\}$$
$$\frac{\partial \gamma}{\partial P_3} = \frac{\partial \mathbf{C}}{\partial P_3}\mathbf{s}^{v}, P_3 \in \{\hat{\alpha}_0, \hat{\delta}_0, \kappa_0, \hat{\alpha}_1, \hat{\delta}_1, \kappa_1, \hat{\alpha}_2, \hat{\delta}_2, \kappa_2, \}$$

We can then proceed to use these partials of γ with respect to the parameters in the observation equations, i.e. for the x and y image measurements (i.e. Colvin's equation 11).

However, note that in a line scanner camera, given the convention here that x is measured along the line (the sample value) and y is measured along the direction of line collection (the line value), the y camera coordinate *will always be zero* because in the formulation here the measures apply only to a given line. The actual geometry of that line is determined from the time of the line *t*, which is derived from the pixel line measurement $line$, e.g.

$$t = t_R + (line - line_R)\delta t_{line}$$

where t_R is the time of the reference line $line_R$ and δt_{line} is the time difference between adjacent lines. For THEMIS IR images, the δt_{line} value should be obtained from the image labels along with the time t_{start} for the starting line. To keep the second order polynomials that will be fitted to the image's position and orientation symmetric along the image, the reference line and time should be chosen to be that of some line in the middle of the image. Therefore

$$t_R = (line_R - line_{start})\delta t_{line}$$

$line_R$ can be chosen somewhat arbitrarily, e.g. as simply the line closest to the middle of the image in question.

Note that this offset in line and time must be tracked so that when updated position and orientation (SPICE) data is output, it is correctly referenced as necessary to the start time of the image.

2.2.4 Parameter Weighting

As noted in Section 2.1 above, it is likely, if not certain, that some of the newly added parameters will need to be constrained (or weighted) if used. For example, it has long been known that for the narrow fields of view commonly used in planetary imaging systems, the orientation of the camera will be highly correlated with the camera (spacecraft) position. This is of course the reason that *randlsq* does not currently allow one to solve for position.

So in order to solve for both position and orientation (and even further for velocity, acceleration, and orientation velocity and acceleration) it will be necessary to weight these parameters. This is done simply by estimating the accuracy of the a priori values of the parameters (for example, we can start by assuming the spacecraft position components are known to an accuracy of 100 meters) and adding the inverse of these estimates (e.g. 1/100 m) to the given parameter's position on the diagonal of the normal equations, before the equations are solved. Such weighting is already possible under various circumstances in *randlsq*, e.g. using the original weighting scheme from RAND, where powers of ten weights could be added to specific parameters, and additions made here, e.g. where MOLA derived ground point positions and radii could be weighted, and where Clementine camera angles could be weighted.

The main changes that will be required in *randlsq* will deal with inputting these weights (as discussed further in section 4 below) and adding them to the normal equations.

It is also obvious that some experimentation will be required in order to determine what weights are needed for a given parameterization. It would also be useful to investigate (e.g. in the case of THEMIS, with those knowledgeable about the 2001 Mars Odyssey mission) how accurate the a priori SPICE information is likely to be and at least start with weights derived from such information.

3. Adjustment Model

Colvin (1992, pp. 6-7) very briefly discusses the overall adjustment model to be used. For completeness and reference, included here are further details on the method of weighted least squares using observations equations. This material derives from Uotila (1986, pp. 61-65, 95). Also see Mikhail and Ackermann (1976) or any number of texts on least square solutions for a similar discussion.

The idealized observation equations are:

$$\mathbf{L}^{\mathbf{a}} = F(X^{\mathbf{a}})$$

where $\mathbf{L}^{\mathbf{a}}$ are the theoretical values of the observations (an *n* x 1 vector, where *n* is the number of observations), which in turn are a function *F* of the theoretical parameter values $X^{\mathbf{a}}$ (a *u* x 1 vector, where *u* is the number of unknowns).

This can also be expressed as:

$$\hat{\mathbf{L}}^{\mathbf{a}} = F(\hat{X}^{\mathbf{a}})$$

where now $\hat{\mathbf{L}}^{\mathbf{a}}$ are the estimated values of the observations, which in turn are the same function *F* of the estimated parameter values $\hat{X}^{\mathbf{a}}$.

$\hat{\mathbf{L}}^{\mathbf{a}}$ can also be expressed as the actual value of the observations, $\mathbf{L}^{b}$ plus a difference vector, known as V, the residuals:

$$\hat{\mathbf{L}}^{\mathbf{a}} = \mathbf{L}^{b} + V$$

The idealized parameter values $\hat{\mathbf{X}}^{\mathbf{a}}$, can also be expressed in terms of their approximate values $\mathbf{X}^{0}$ plus a difference vector $\hat{\mathbf{X}}$:

$$\hat{\mathbf{X}}^{\mathbf{a}} = \mathbf{X}^{0} + \hat{\mathbf{X}} \qquad (8)$$

If we take the partials of the function *F* with respect to the parameter values, we obtain the partials:

$$\frac{\partial F}{\partial \hat{\mathbf{X}}^{\mathbf{a}}} = \mathbf{A} \qquad (9)$$

where $\mathbf{A}$ is an *n* by *u* matrix.

We also can also use the approximate parameter values $\mathbf{X}^{\mathbf{0}}$ in the function *F*, in order to derive initial estimates $\mathbf{L}^{\mathbf{0}}$ of the observations:

$$\mathbf{L}^{\mathbf{0}} = F(\mathbf{X}^{\mathbf{0}}) \qquad (10)$$

The difference between these initial estimates of the observations $\mathbf{L}^0$ and the observations themselves $\mathbf{L}^b$ is:

$$\mathbf{L} = \mathbf{L}^0 - \mathbf{L}^b \qquad (11)$$

(As an aside, the negative of the above values are often called the "O-C's", i.e. the observed minus the computed values.)

We also can estimate in various ways a weight matrix for the observations, i.e.:

$$\mathbf{P} = \sigma_o^2 \sum_{\mathbf{L}^b}^{-1} \qquad (12)$$

where $\mathbf{P}$ is an *n* by *n* weight matrix, σ_o^2 is the *a priori* variance of unit weight (usually set to 1), and $\sum_{\mathbf{L}^b}^{-1}$ is an *n* by *n* estimated weight matrix for the observations. This is usually a diagonal matrix, with the diagonal elements only expressing the estimated relative accuracy of the observations. As we will see in our case the observations are all image measurements, so they may indeed all have the same relative accuracy, meaning $\sum_{\mathbf{L}^b}^{-1}$ could be an identity matrix.

The normal (*u* by *u*) matrix then becomes:

$$\mathbf{N} = \mathbf{A}^T\mathbf{P}\mathbf{A} \qquad (13)$$

It can then be shown (using the method of minimizing the sum of the squares of the residuals $\mathbf{V}$) that the correct solution for the parameters' differences $\hat{\mathbf{X}}$ is given by the following normal equations:

$$\hat{\mathbf{X}} = -(\mathbf{N})^{-1}\mathbf{A}^T\mathbf{P}\mathbf{L} \qquad (14)$$

The residuals then are:

$$\mathbf{V} = \mathbf{A}\mathbf{X} + \mathbf{L} = (\mathbf{P}^{-1} - \mathbf{A}\mathbf{N}^{-1}\mathbf{A}^T)\mathbf{P}\mathbf{L} = \mathbf{Q}_v\mathbf{P}\mathbf{L} \qquad (15)$$

The (scalar) sum of the square of the residuals is:

$$\mathbf{V}^T\mathbf{P}\mathbf{V} = \mathbf{L}^T\mathbf{P}\mathbf{L} + \hat{\mathbf{X}}^T\mathbf{A}^T\mathbf{P}\mathbf{L} \qquad (16)$$

The (scalar) *a posteriori* variance of unit weight is:

$$\hat{\sigma}_o^2 = \frac{\mathbf{V}^T\mathbf{P}\mathbf{V}}{n-u} \qquad (17)$$

where as before, *n* is the number of observations and *u* is the number of unknown parameters. "*n-u*" is also known as the *degrees of freedom* of the solution.

It can also be shown that the estimated variance-covariance matrix (of size *u* by *u*) for the parameters, assuming normally distributed errors in the measurements, is:

$$\sum_{\hat{\mathbf{X}}^a} = \sigma_o^2(\mathbf{N})^{-1} \qquad (18)$$

The estimated variance-covariance matrix for the idealized observations (of size *n* by *n*) then is:

$$\sum_{\mathbf{L}^a} = \sigma_o^2\mathbf{A}(\mathbf{N})^{-1}\mathbf{A}^T \qquad (19)$$

This can be compared to $\sum_{\mathbf{L}^b}$ in order to make sure the initial estimate of weights for the observations is reasonable.

The estimated variance-covariance matrix (of size *n* by *n*) for the residuals is sometimes desired, and it is:

$$\sum_{\mathbf{V}} = \sigma_o^2(\mathbf{P}^{-1} - \mathbf{A}(\mathbf{N})^{-1}\mathbf{A}^T) \qquad (20)$$

4. Implementation

Initially it was not clear whether the adjustment of line scanner camera images should be implemented as changes in the RUPG software (i.e. *randlsq*), in a totally new program, or in software that is to be developed for DISR and MER image processing (Archinal, 2004a).

However, at this point in time, it appears that the first option should be followed. This appears to be the fastest way to reach an implementation which we can start to use for testing purposes. It will rely on already well tested and working algorithms and I/O procedures. It will also facilitate the combined adjustment of image measurements from framing cameras and line scanner cameras – which we need to do as part of this project (e.g. using the THEMIS IR line scanner camera images along with framing camera images, such as from Viking or THEMIS VIS).

There would seem to be little to no advantage in writing a totally new program given that it seems the algorithm listed above will require some fairly straightforward (but admittedly extensive) changes to the *randlsq* program. We have also not progressed far enough along with a planned adjustment program for DISR and MER and cannot wait (the likely few more months at least) for such software to be written and tested, before adding a line scanner camera model adjustment.

So it appears that we should proceed to modify the current version of *randlsq* (currently available as laxmi:/work1/barchina/rupg/src/randlsq-ba-dev/randlsq.F). This is written in generic Fortran. I personally would prefer Fortran continue to be used for changes (both in the interest of time and so I can assist with changes now and in the future if desired(!)), but we should consider whether parts or all of the code should be rewritten in C (or even C++?), if it would speed up the overall project or be planned for near-term work (e.g. under the ISIS IM project). Some existing ISIS routines will likely have to be called, so changes to assist in calling such (C) routines will be necessary in any case.

Bug fixes and some minor development of the *randlsq* program will probably continue even after the modifications described here are started. Once this major modification of *randlsq* is complete, all of the changes should be merged together into one compatible program.

Following are subsections which consider the various types of changes that would be needed to the randlsq.F (or its converted equivalent) code.

4.1 Input/Output Changes

Some changes will be necessary to read the THEMIS IR specific image measurements, as well as to read the SPICE data for such images. Some changes will also be necessary in the "program control" input, and whether those are simply added to the current input or the input of this information is revamped is a matter of efficiency open to discussion.

4.1.1 Image Measurements

The image measurements will have been made by ISIS *qmatch* (or any equivalent software) and will be in pixel form. These will have to be pre-converted or read in and converted into mm measurements along each line, and time measurements along the image. Camera model information will also have to be used in a pre-processing step or after reading the pixel measurements, e.g. regarding focal length and image distortions along each line.

So, to be specific, the choice should be made whether to convert the pixel measurements in a pre-processing program (to an x value in mm and a y value in

time) or to convert them while reading them. If the latter, for line-scanner image measurements, the current *randlsq* reading of pixel measurements would be bypassed, and input code written to read this information and needed information from the images, and to do conversions as necessary.

In either case, the along line pixel measurements should be converted to mm, using the appropriate focal length (image scale) information, and camera model (taking care to use the values for the correct filter if they differ). This information would have to hard coded, or more preferably read from either the images or the ISIS THEMIS parameters file ($ISISM01DATA/THM_parameters.def.7). The along track pixel measurements should be converted to time, using the start time of the image (reference time), and the line time interval of the image (keyword "IRLINERATE"), as read from the labels of the images or the just mentioned parameter file. During the solution itself, these times can then be converted to time differences from the middle of the image, as indicated in Section 2.2.3 above.

4.1.2 SPICE Data

Currently *randlsq* reads SPICE data (in a specific ASCII format) for each image, including the time (Julian date), the camera orientation, and spacecraft position. For THEMIS IR and future line scanner cameras, this SPICE information is much more extensive, with the information available (and changing) several times in the course of the collection of a single image. Rather than try to emulate the current *randlsq* program method (e.g. by putting out for each image these e.g. 7 numbers many, many times in ASCII format!), it would seem far more reasonable to use (e.g. ISIS or NAIF) routines to read the SPICE data directly. Similarly, once this information is updated via a solution, it should be written out so it will be available for further ISIS processing.

So if line scanner image data were being read, the reading of the SPICE data from the current a priori file (sometimes called the pole, points, and position file) would be bypassed, and the SPICE data read in directly using the appropriate ISIS or NAIF routines.

Note that the "pole and points" information could be (should be?) still read from the same file as before, because this same information is needed as with any framing camera solution.

Also note that care must be taken regarding the definitions of the rotations as obtained from the SPICE CK kernel, since they may require transformation to match the rotation angle system used in *randlsq*.

After a solution is complete, rather than outputting the SPICE data to an updated version of the a priori (poles, points, and position) file, a separate binary file of SPICE information should be output using appropriate NAIF/ISIS routines.

4.1.3 Program Control (Parameters) Information

Currently, *randlsq* reads all program control information from a single control (or "parameters") file. Additional information will be needed to direct the *randlsq* program actions when handling line scanner camera images. Specifically, the location of the input and output SPICE files, which additional parameters should be solved for on additional images, and the weights for those parameters, will all need to be read in.

As noted above, this could probably be done by simply extending the format of the current parameters files, particularly in regard to specifying the additional parameters and their weights to be solved for. The SPICE files names would have to be read from e.g. new lines in the parameter file, or from direct or command line entries for the program.

An alternative might be to abandon the current method of reading the parameters entirely, and instead adopt the read of a PVL file, such as that set up by Ben Atkins, and used in his *runrandlsq.pl* script. See his web pages at (e.g. for now) Magnus:/home/batkin/projects/rupg/`uname -s`/docs/html .

4.2 Computational Changes

These changes are the heart of the whole process of handling line scanner camera images. The essential changes are described in Section 2 above, by outlining the additional parameters that we need to solve for for each image, and the additional partial derivatives that need to be computed to allow for the computation of these parameters. I am not going to try to describe these changes in detail (e.g. line by line in the program) but will try to outline them here.

4.2.1 Generation of A priori Information

Since we are solving for additional parameters, we need to also generate a priori values for those parameters. The additional parameters are the second order polynomial parameters in position and orientation. The easiest way to do this is to successively treat the three elements of position and three elements of orientation, by taking the SPICE information for each of these (SPK and CK respectively) and fitting second order polynomials to this SPICE data. (The SPICE could be evaluated at each line or perhaps more efficiently, at the position of each position (in the line direction) where a measurement exists, and the polynomials fitted to the resulting values.) This curve fitting should be done in some standard way, e.g. with some standard routines (e.g. Numerical Recipes) that I will not try to describe here. These a priori parameter values can then be used to generate the "computed" values of the "observations" (the mm sample

measurements, and time (line) measurements), as well as the desired partial derivatives.

CK kernels for 2001 Mars Odyssey exist as *.bc files in the $ISISM01DATA directory or the NAIF ftp site (e.g. “m01_sc_map9.bc”). These are NAIF “type 3” CK kernels (Bachman and Elson, 2002a), where pointing and the rate of change of pointing is given for discreet epochs over given intervals (e.g. when an image is being collected, or according to the kernel comments, when there is data more often than every 60 seconds). Note however that kernel comments indicate the (angular) velocity information has simply been interpolated from the orientation information, so it does not really provide any additional information. The naiflib CK kernel access software will linearly interpolate this orientation and orientation rate information at any times requested. Initially it would probably be simplest to use the appropriate naiflib routines to provide this information before a polynomial fit is made to it in randlsq, but eventually it might be worth testing whether the information should be read for the specific times it is available, and that information fitted to a polynomial directly.

SPK kernels for 2001 Mars Odyssey exist as *.bsp files in the $ISISM01DATA directory of the NAIF ftp site (e.g. “m01_map9.bsp”). These are NAIF “type 1” PCK kernels (Bachman and Elson, 2002b), i.e. with “modified difference arrays” (MDAs). This information appears to provide spacecraft position and velocity at discrete intervals. Again, it would probably be simplest to use appropriate naiflib routines to collect this information and generate a priori polynomial coefficients, but it might be useful to test at some point whether using the MDAs directly would provide an increase in accuracy.

4.2.2 Partial derivatives

Partial derivatives will need to be computed as described in Section 2 above. Note here (and for the a priori computation) that for some images the use of all the possible new parameters will be optional. In other words, even if we do so for test purposes, not all parameters will be solved for every image, so additional bookkeeping to keep track of the parameters to be solved for for each image will have to be added.

The basic idea is that the *randlsq* software already solves for partials for most of the parameters of interest. Also these partials are often solved for by the chain rule, so we need only solve for the very specific partials (listed in Section 2) for the new parameters, and then continue to compute the final partials (of the observations with respect to the parameters) by continuing to apply the chain rule, using some partials that are already available. (I know, I know, this may not be entirely clear, but again, rather than cover this now line by line in randlsq.F, we can discuss it and work though it and the code if necessary.)

Various changes to handle the “overhead” of additional parameters will also be needed, e.g. dimensioning the initial arrays large enough, keeping track of which new parameters are needed for each image, and keeping track of weights (if any) on the new parameters. Output will also need to be added to show the a priori and adjusted values of the new parameters for each image.

Additionally, I have found it useful in least squares adjustment programs to keep track of the list of parameters being solved for in one (character) array. This at least facilitates debugging and error checking in that values such as solved for parameters and their corresponding normal equation values can be checked and output as needed and quickly associated with their real meaning (as opposed to just their parameter number). This type of tracking was not in the RAND software originally, as it simply recomputed the position of a parameter in the normal equations every time it needed it. However, I have partially added this information in the form of the array “namep”, and this usage could easily be extended to handle the new parameters (handling not only a name indicating the type of parameter as is currently done, but also the image or point name that it’s associated with).

4.2.3 Solution Output

This has mostly been covered above, but after a solution has converged, additional output related to the line scanner camera data will be needed.

At the least, “printed” output will be needed to show the a priori and solved for values, and uncertainties, of any new parameters added, e.g. the solved for polynomial coefficients for each image. The uncertainties are only accessible in matrix inversion solutions, but with small numbers of images we should be able to do all the solutions by this method (handling a global mosaic will require the use of a conjugate gradient solution and/or the use of true block adjustment solution techniques.) Along these same lines, it will also probably be quite useful to see the correlations (derived from the covariances, i.e. the off axis elements of the inverted normal matrix) of the position and orientation parameters for each image. This would help to verify what types of weights are needed on these parameters and which parameters are even meaningful to solve for (since we know the correlation will be high between the position and orientation).

The parameter weights actually used should also be shown with this output, or perhaps more conveniently to program, when they are set before the start of the solution (as they are now in a couple of different forms).

We will also need to go back from the solved for polynomial coefficients to the representation used in the SPICE data. So the reverse procedure of that described above will now be needed. The second order polynomials will be used to evaluate the position, velocity, and orientation information, e.g. at each line, and then this new information could be output using the appropriate naiflib

routines for position and velocity ("type 1" SPK) and orientation ("type 3" CK). Further rotations (the reverse of any applied during reading) may need to be applied to the CK information before output, in order to convert from the *randlsq* to the CK file rotation system.

This should complete the coverage of the changes needed. The essential output of the solution will be represented by the updated SPICE kernels, from which new, geodetically controlled, mosaics can be generated, using normal ISIS procedures. Updated positions for the control points will also be available as usual from the *randlsq* output.

5. Testing and Near Term Work

Tie point measurements on THEMIS IR images are being collected by L. Weller in the area of the 2 MER landing ellipses. For at least one pair of mostly overlapping images, a substantial set of such measures is being collected (e.g. ~100? tie points), and for the others reasonable numbers of tie points (e.g. ~10 per image pair, including measurement against MOLA DIMs).

Initial testing of the software should therefore center on adjusting the geometry of the pair of overlapping images with a dense set of measures. An initial solution should be done using the same set of parameters normally used for a framing camera solution, i.e. solving only for the (3 angle) orientation of one and/or both of the images. Successive solutions should then be done, increasing the number of parameters added in each solution (e.g. first adding position, then velocity of orientation change, then spacecraft velocity, then acceleration of orientation change, and finally spacecraft acceleration) and adjusting weights on these parameters if appropriate. A similar series of solutions should also be done, reducing the number of tie points in each in order to see where the "break even" point is for tie points per image, for the given parameterization. After each solution or sets of solutions, an analysis can be done to see if the addition of the parameters in question significantly improved the solution or not. The residuals can also be examined to see if there is some motion that is not being adequately compensated for by the use of the second order polynomial fits.

After this, the solutions should shift to handling the multiple images covering the MER landing ellipses, redoing some of the same sorts of comparisons indicated above. In particular, the absolute positions of some number of points located in the adjustment should be compared with their MOLA DIM measured locations.

A final step in this (CDP funded portion of the) project will be to collect tie point measurements from THEMIS VIS images, and adjust this data along with the corresponding THEMIS VIS image measures. Some additional programming will be needed here in order to preprocess the VIS image measures, e.g. as is done for the other framing camera cases (e.g. a short preprocessing program like

isschange will be needed to convert pixel measures to idealized mm measures). But this will assure that VIS and IR image data can be adjusted together, and for that matter more generally that randlsq could then be used to adjust any framing camera measures (e.g. including Mariner 9 and Viking) along with THEMIS IR measures.

If time permits (or in later work), we should also attempt to add as a parameter an offset to the absolute start time of each image or image portion. The latter case will address the situation of images were lines are missing and the absolute time of some set of lines is not known. The former case will address the situation where the difference between the commanded and actual start time of an image is not known. This parameter could only be solved for relative to absolute (i.e. MOLA based) ground control, or if overlapping descending and ascending images (day and night) were adjusted together.

Another addition that should be added fairly early on is the automatic handling of images that are "broken", i.e. have any missing lines. If such an image is processed (recognized how from the labels?), it should be treated as two different images, each with its own set of parameters. It may be necessary to reject (with a suitable error message) the processing of any such partial images where insufficient measures then exist for the different portions. A further extension of this would be to add an option to allow the user to manually treat some images in such a fashion, e.g. when other than automatically available information indicates the image was "broken".

6. Future Work

Other, somewhat longer term issues are what additional parameters will need to be added (if any) for camera calibration, and also what additional types of constraints on the total transformations are needed.

At some point in the future, it might be useful to add additional capabilities for the THEMIS IR line scanner camera system, e.g. for calibration purposes, and in order to treat the multiple infrared bands collected during each image acquisition as a multi-line scanner camera system. For calibration, we should at least try to allow for the adjustment of the line scanner position (in the THEMIS case, as integrated over the lines used for each filter) relative to the camera boresight. Substantial information on this is available in the THEMIS frames kernel, e.g. locally at magnus:/home/jtorson/ik/m01_themis_v31.ti or from the NAIF ftp site. (Also see the 2001 Mars Odyssey frames kernel, available as $ISISM01DATA/m01_v26.tf or from the NAIF ftp site.) Tom Duxbury and Anton Ivanov (both at JPL) have also done some investigations concerning these parameters and could be consulted for additional information.

It may also be desirable to add additional constraints or to even change the set of "additional parameters" described here in order to better solve for the line scanner camera geometry. Only by testing using the procedures here will we likely see whether such improvements would be worthwhile.

It would also be useful to add an option to not only break images into pieces and treat them separately (as discussed in the previous section) but to optionally make sure such images are treated in a piecewise continuous fashion if there in fact are no missing lines between the pieces. Here constraints should be placed on the position, velocity, orientation, and orientation velocity so that the last line of one image will be handled in a consistent way with the next line – the first line of the next image.

As noted already, eventually we plan to add capabilities to process data from the other planetary line scanner camera systems (MOC, HRSC, and HiRISE). Additional parameterization may also be needed for camera calibration of these instruments as well. See for example R. Kirk's summary of a camera (sensor) model for HiRISE (Appendix II below), where various camera calibration constants are described.

Acknowledgements

This work has relied on funding from the NASA Critical Data Products initiative. Thanks to Jim Torson for information on parameters relevant to the THEMIS camera (personal communication of 2004 December 3) and to Randy Kirk for the information on HiRISE camera modeling.

References

The availability of a reference is often indicated following the reference. Comments on the material covered are in the set of brackets. Not all of these references are currently cited above.

Archinal, B. A., et al. (2000), A New RAND-USGS Control Network and Size Determination for Io, *Eos Trans. AGU (suppl.)*, 81, p. F317. [Io control. RUPG software.]

Archinal, B. A., et al. (2001), An Improved RAND-USGS Control Network and Size Determination for Io, *Lunar Planet. Sci., XXXII,* Abstract 1746, Lunar and Planetary Institute, Houston (CD-ROM). [Io control. RUPG software.]

Archinal, B. A., et al. (2002), A MOLA-Controlled RAND-USGS Control Network for Mars, *Lunar Planet. Sci., XIII*, Abstract no. 1632, Lunar and Planetary Institute, Houston (CD-ROM). [MDIM 2.1. RUPG software.]

Archinal, B. A., et al. (2003a), Mars Digital Image Model 2.1 Control Network, *Lunar Planet. Sci., XIVI*, Abstract no. 1485, Lunar and Planetary Institute, Houston (CD-ROM). [MDIM 2.1. RUPG software.]

Archinal, B. A., et al. (2003b), Mars Digital Image Model (MDIM) 2.1 Control Network, in ISPRS WG IV/9: Extraterrestrial Mapping Workshop *Advances in Planetary Mapping 2003,* Houston, TX. Available at http://astrogeology.usgs.gov/Projects/ISPRS/MEETINGS/Houston2003/final.html. [MDIM 2.1. RUPG software.]

Archinal, B. (2004a), Modeling and Adjustment of Mars Exploration Rover Camera and Coordinate Systems Geometry, April 10, unpublished. [Partial basis for this document. Describes planned adjustment for DISR, MER, and perhaps eventually all planetary images.]

Archinal, B. A. (2004b), Documentation for RAND-USGS Planetary Geodesy Software. April 14, unpublished. [RUPG (or alternately, RUPICON) software.]

Archinal, B. A., Lee, E. M., Kirk, R. L., Duxbury, T. C., Sucharski, R. M., Cook, D. A., and Barrett, J. M. (2004c), A New Mars Digital Image Model (MDIM 2.1) Control Network, *Commission IV, WG IV/9, XXth ISPRS Congress*, 12-23 July, Istanbul, Turkey. Available at http://www.isprs.org/istanbul2004/comm4/papers/464.pdf. [MDIM 2.1, Mars control.]

Bachman, N. J., and L. S. Elson (2002a), CK Required Reading, (Navigation Ancillary Information Facility, Jet Propulsion Laboratory, Pasadena, CA). September 5. NAIF Document 174.08. Available from the NAIF ftp site. [Description of NAIF CK kernels.]

Bachman, N. J., and L. S. Elson (2002b), SPK Required Reading, (Navigation Ancillary Information Facility, Jet Propulsion Laboratory, Pasadena, CA). September 5. NAIF Document 168.10. Available from the NAIF ftp site. [Description of NAIF SPK kernels.]

Christensen, P.R., J.L. Bandfield, J.F. Bell, III, N. Gorelick, V.E. Hamilton, A. Ivanov, B.M. Jakosky, H.H. Kieffer, M.D. Lane, M.C. Malin, T. McConnochie, A.S. McEwen, H.Y. McSween, Jr., G.L. Mehall, J.E. Moersch, K.H. Nealson, J.W. Rice, Jr., M.I. Richardson, S.W. Ruff, M.D. Smith, T.N. Titus, and M.B. Wyatt (2003), Morphology and composition of the surface of Mars: Mars Odyssey THEMIS results, *Science*, 300: 2056–2061. [THEMIS.]

Ebner, Heinrich, Michael Spiegel, Albert Baumgartner, Bernd Giese, Gerhard Neukum, and the HRSC Co-Investigator Team (2004), Improving the Exterior Orientation of Mars Express HRSC Imagery, *Commission IV, WG IV/9, XXth*

ISPRS Congress, 12-23 July, Istanbul, Turkey. http://www.isprs.org/istanbul2004/comm4/papers/462.pdf. [Adjusts bias to phi and kappa only, or uses MOLA DEM points for control and also adjusts for single position and omega offset. Automatic tie point matching. HRSC processing.]

Eliason, E. M. (1997), Production of Digital Image Models Using the ISIS System, Lunar and Planetary Institute Conference Abstracts, *28th Annual Lunar and Planetary Science Conference*, March 17-21, 1997, Houston, TX, p. 331. [General reference for ISIS.]

Fritsch, Dieter, and Dirk Stallmann (2002), Rigorous Photogrammetric Processing of High Resolution Satellite Imagery, *IAPRS Vol. XXXIII, Part B1, Comm. I, pp. 313-321, ISPRS Congress*, Amsterdam. 2000. Available at http://www.ifp.uni-stuttgart.de/publications/2000/Stallmann_1576.pdf. [Highly derivative of Kratky (1989). Polynomial representation of position and orientation changes during image acquisition. Use of Gauss-Helmert adjustment model (observations and conditions). SPOT, IRS-1C, and MOMS-2P/PRIRODA processing.]

Gaddis, L.; Anderson, J.; Becker, K.; Becker, T.; Cook, D.; Edwards, K.; Eliason, E.; Hare, T.; Kieffer, H.; Lee, E. M.; Mathews, J.; Soderblom, L.; Sucharski, T.; Torson, J.; McEwen, A.; Robinson, M. (1997), An Overview of the Integrated Software for Imaging Spectrometers (ISIS), *28th Annual Lunar and Planetary Science Conference*, March 17-21, 1997, Houston, TX, p. 387. [General reference for ISIS.]

Gruen, A., and Z. Li (2001), TLS Data Processing Modules, PowerPoint presentation, *3rd International Image Sensing Seminar on New Development in Digital Photogrammetry*, 24-27 September, Gifu, Japan. Available at http://www.chikatsu-lab.g.dendai.ac.jp/wgv4/presentation/08_01Li.pdf. [A good general presentation on line scanner camera systems. Describes basic models for handling line scanner cameras, i.e. a) Direct georeferencing (DGR); b) Piecewise polynomial model (RPM); and c) Cubic spline interpolation model (CSI). TLS System. Image matching. Automatic matching in a multi-line scanner camera.]

Gruen, A., and Z. Li (2002), Automatic DTM Generation from Three-Line-Scanner (TLS) Images, *GIT Kartdagar Symposium*, 17-19 April, Stockholm. . Available from http://www.photogrammetry.ethz.ch/research/TLS/pub/automatic_DTM_tls_stockholm.pdf. [Automatic matching in a multi-line line scanner camera. TLS System.]

Gruen, A., and Z. Li (2003), Sensor Modeling for Aerial Triangulation with Three-Line-Scanner (TLS) Imagery, *Journal of Photogrammetrie, Fernerkundung, Geoinformation*, 2, pp. 85-98. Available at

http://www.photogrammetry.ethz.ch/general/persons/zhangl_pub/TLS_PFG.pdf. [Automatic matching in a multi-line scanner camera system. TLS System. This is a more detailed version of Gruen and Li (2002) above.]

Kratky, V. (1989). Rigorous photogrammetric processing of SPOT images at CCM Canada, *ISPRS Journal of Photogrammetry and Remote Sensing*, 44, 53-71. [Polynomial representation of position and orientation changes during image acquisition. SPOT processing.]

Malin, M. C., G.E. Danielson, A.P. Ingersoll, H. Masursky, J. Veverka, M.A. Ravine, and T.A. Soulanille (1992), Mars Observer Camera, *J. Geophys. Res., 97*: 7699–7718. [MOC.]

Malin, M.C., and K.S. Edgett (2001), Mars Global Surveyor Mars Orbiter Camera: Interplanetary cruise through primary mission, J. Geophys. Res., 106(E10): 23,429–23,570. [MOC.]

McEwen, A. S., Delamere, W. A., Eliason, E. M., Grant, J. A., Gulick,, V. C., Hansen, C. J., Herkenhoff, K. E., Keszthelyi, L., Kirk, R. L., Mellon, M. T., Squyres, S. W., Thomas, N., and Weitz, C. (2002), HiRISE: The High Resolution Imaging Science Experiment for Mars Reconnaissance Orbiter, *Lunar Planet. Sci., XXXIII*, Abstract #1163, Lunar and Planetary Institute, Houston (CD-ROM). [HiRISE.]

Neukum, G., H. Hoffmann, R. Jaumann, and the HRSC Co-Investigator Team (2004), The High Resolution Stereo Camera (HRSC) Experiment on the ESA Mars Express mission. *Int. Arch. Photogramm. Remote Sens., XXXV*, B, "Geo-Imagery Bridging Continents", Istanbul, DVD-ROM. [HRSC.]

Mikhail, Edward M., and F. Ackermann [1976]. *Observations and Least Squares*, (Harper and Row, Publishers, New York). [Basic reference on least squares adjustment. Uotila (1986) is more useful, but not generally available.]

McGlone, J., ed. (2004), *Manual of Photogrammetry Fifth Edition*, American Society for Photogrammetry and Remote Sensing, Bethesda, Maryland. Available from the USGS Flagstaff library as 753.5 Am3sm 2004. [Basic reference on photogrammetry.]

Montenbruck, Oliver, Eberhard Gill, and Timm Ohlhof (1994). A Combined Approach for Mars-94 Orbit Determination and Photogrammetric Bundle Adjustment, *DLR Forschungsbericht* 94-13. [Copy available from Randy Kirk.] [Assumes fairly rigorous photogrammetric processing of HRSC data, including use of such data to rigorously improve orbit, and even improve Mars rotation information. Explanation of this complex model is useful, but it is mostly made obsolete by the availability of MOLA data. HRSC processing.]

Oberst, J. (2002), Orbit and Pointing Data Adjustment for HRSC on Mars Express (HWBUNDLE *light*), PowerPoint presentation, presented at the October HRSC Team meeting. [Solving for orbit parameters (position and velocity) and orientation, the latter parameterized via Fourier series so that "jitter" can b recovered. HRSC processing.]

Oberst, J., T. Roatsch, B. Giese, M. Wahlish, F. Scholten, K. Gwinner, K.-D. Matz, E. Hauber, R. Jaumann, J. Albertz, S. Gehrke, C. Heipke, R. Schmidt, H. Ebner, M. Spiegel, S. vanGasselt, G. Neukum, and the HRSC Co-Investigator Team (2004), The Mapping Performance of the HRSC/SRC in Mars Orbit, *Commission IV, WG IV/9, XXth ISPRS Congress*, 12-23 July, Istanbul, Turkey. Available at http://www.ipi.uni-hannover.de/html/publikationen/2004/paper/oberst_etal_04_istanbul.pdf. [HRSC and SRC processing.]

Poli, Daniela (2004), Orientation of Satellite And Airborne Imagery From Multi-Line Pushbroom Sensors With A Rigorous Sensor Model, *Commission I, WG I/5, XXth ISPRS Congress*, 12-23 July, Istanbul, Turkey. Available from http://www.isprs.org/istanbul2004/comm1/papers/25.pdf. [Good brief reference. Position and attitude modeled rigorously with second order polynomials. Alternative of using the rational polynomial coefficients (RPC) also presented. Automatic tie pointing. SPOT-5/HRS processing.]

Poli, Daniela, Zhang Li, and Armin Gruen (2004), SPOT-5/HRS Stereo Images Orientation and Automatic DSM Generation, *Commission I, XXth ISPRS Congress*, 12-23 July, Istanbul, Turkey. Available at http://www.isprs.org/istanbul2004/comm1/papers/77.pdf. [Position and attitude modeled with piecewise second order polynomial functions, with trajectory divided into segments based on available points. Self calibration parameters. Also describes tests with rational functions (RPC or rational polynomial coefficients) model. Automatic tie pointing. SPOT-5/HRS processing.]

Schmidt, R., and R. Brand (2003), Automatic Determination of Tie Points for HRSC on Mars Express, *Proceedings Joint ISPRS/EARSeL Workshop "High Resolution Mapping from Space 2003*, Institut für Photogrammetrie und GeoInformation, Universität Hannover, 6 p, on CD ROM. Available at http://www.ipi.uni-hannover.de/html/publikationen/2003/workshop/schmidt.pdf. [Automatic tie point matching. MOMS-02/D2, MOMS-2P, HRSC-A, and HRSC processing.]

Slama, Chester C. (ed.) (1980), *Manual of Photogrammetry, Fourth Edition*, (American Society of Photogrammetry, Falls Church, VA). [Basic reference on Photogrammetry. Now mostly superseded by McGlone (2004).]

Torson, J. M.; Becker, K. J. (1997), ISIS - A Software Architecture for Processing Planetary Images, 28th Annual Lunar and Planetary Science

Conference, March 17-21, 1997, Houston, TX, pp. 1443-1444. [Basic reference for ISIS.]

Toutin, Thierry (2004), Review Paper: Geometric processing of remote sensing images: models, algorithms and methods, *International Journal of Remote Sensing*, 25(10), pp.1893-1924, http://www.ccrs.nrcan.gc.ca/ccrs/rd/sci_pub/bibpdf/13288.pdf (accessed 9 May 2004). Preprint available at http://geomatica.unipv.it/autec/Toutin-1.pdf. [General reference on photogrammetric processing, including use of line scanner cameras. Section 3 has a good review of methods used so far to process line scanner images.]

Uotila, Urho A. (1986). *Notes on Adjustment Computations Part I*, (Department of Geodetic Science and Surveying, The Ohio State University, Columbus, Ohio). [Includes good description of least square adjustment models. Not generally available, but copy available from Archinal.]

Zoej, M. J. Valadan, and S. Sadeghian (2003), Rigorous and Non-Rigorous Photogrammetric Processing of Ikonos Geo Image, Joint Workshop of ISPRS Working Groups I/2, I/5, IC WG II/IV, and EARSeL Special Interest Group: 3D Remote Sensing. Available at http://www.ipi.uni-hannover.de/html/publikationen/2003/workshop/valadan.pdf. [Lists "rational functions", "direct linear transformation" (DLT), "self calibration DLT", "3D affine transformation" models. Describes "orbital parameter" model, using Keplerian parameters, and adjusting 3 orbit orientation angles with polynomial expressions. Ikonos processing (i.e. where no sensor model is available).]

Useful web sites

ISIS, available at
http://isis.astrogeology.usgs.gov/

Three Line Scanner Project, available at
http://www.photogrammetry.ethz.ch/research/TLS/TLS_Publications.html.

Appendix I

Partial Transcription of Colvin (1992) Documentation

= = = = = = = = = =

[cover]

A RAND NOTE

Photogrammetric Algorithms and Software for Spacecraft Optical Imaging Systems

Tim R. Colvin

RAND

[inside cover]

The research described in this report was sponsored by the Jet Propulsion Laboratory under Contract No. 958723.

Published 1992 by RAND
1700 Main Street, P.O. Box 2138, Santa Monica, CA 90407-2138

[page i]

A RAND NOTE

Photogrammetric Algorithms and Software for Spacecraft Optical Imaging Systems

Tim R. Colvin

Prepared for the
Jet Propulsion Laboratory

RAND

[page ii is blank]

[page iii]

PREFACE

This Note describes photogrammetric algorithms and software that can be used with data obtained from spacecraft optical imaging systems. The algorithms and software allow for control point coordinates, camera orientation angles, the position of a target body's north pole, and the target body's rotation rate.

This work can be used to generate control point networks and ancillary data for target bodies, which can then be used to mosaic individual photographs into unified cartographic products. Hence, this work will be useful to many analysts in the planetary science community.

This Note is intended to be used as a brief reference, rather than an all-encompassing dissertation on the subject. The intended audience consists of planetary scientists familiar with the basic data and possessing an intimate knowledge of FORTRAN.

This work was done as part of the Galileo project for the Jet Propulsion Laboratory. It has been tested with early data from Galileo, viz., the encounter with the asteroid Gaspra, and with older Voyager data.

[page iv is blank]

[page v]

SUMMARY

Photographic images taken by a spacecraft, and ancillary data for these images, are used to solve for control point variables (latitude, longitude, and radius), camera orientation angles (right ascension, declination, and twist), and target body pole variables (right ascension and declination of the north pole, and rotation rate).

This is done by first making pixel measurements on the images and transforming them to coordinates in the camera focal plane. Then, for the time at which an image was taken, spacecraft position, camera orientation angles, the position of the target body's north pole, the target body's rotation rate, control point position, and camera focal length are used to calculate corresponding coordinates in the camera focal plane. The observed and calculated coordinates are compared in a least-squares adjustment that solves for the above variables.

The mathematics of the photogrammetric least-squares procedure is described in detail. It is presented in such a manner that readers desiring to make modifications are accommodated.

The FORTRAN programs GALGAUSS and GALCGRAD are presented. They solve the normal equations resulting from the least-squares adjustment in different ways. One program employs Gaussian elimination, and the other uses the conjugate gradient method to cleverly solve large systems with minimal storage.

The appropriate input and output files for the two FORTRAN programs are discussed and the programs themselves are extensively commented.

[page vi is blank]

[page vii]

CONTENTS

[page viii is blank]

[page 1]

1. Introduction

Let's say I have in front of me a collection of images of some target body, e.g., the Moon, taken by a spacecraft with an optical imaging system. And let's say I want to deduce some things from these images. Specifically, I would like to know the latitude, longitude, and radius of certain features on the target body. These features are called control points. I would also like to know where the target body's north pole lies and how fast the body rotates. Finally, I want to

know the precise direction the spacecraft's camera was pointing when it acquired an image.

These tasks are accomplished by first making pixel measurements on the images of the features in which I am interested. Because I know the focal length of the camera, I can transform these measurements to coordinates in the camera's focal plane. These are the observational data. I have obtained them using just the images and by knowing the focal length.

Remember our objectives from the first paragraph. I solve for these variables by comparing the observational data, in a least-squares fashion, with a set of calculated data. The calculated data are derived, in part, from approximations for the following variables: control point location (latitude, longitude, and radius), camera orientation angles (right ascension, declination, and twist), and target body pole variables (right ascension and declination of the north pole, and rotation rate). Two other quantities are necessary to obtain our set of calculated data. These are the times at which the images were taken and the positions of the spacecraft at those times.

The least-squares fit of the calculated data to the observational data yields improvements to the approximated variables. I can continue to iterate in this manner until all variables converge. At this juncture I have a set of data that is self-consistent.

Section 2 presents the mathematics of the photogrammetric least-squares adjustment. I have attempted to be rigorous as well as general, so that a reader wishing to make modifications will have an easy time doing so.

Section 3 describes the programs GALGAUSS and GALCGRAD (program listings are in the appendices) and their inputs and outputs. Sample input and output files are given to demonstrate the formats of these files.

[page 2]

2. PHOTOGRAMMETRIC ALGORITHMS FOR SPACECRAFT OPTICAL IMAGING SYSTEMS

We begin with a collection of photographic images. Every image has associated with it a set of control points. Each control point on an image is coupled with a pair (x_0, y_0), measured on the image and transformed to coordinates in the camera focal plane, and a latitude, longitude, and radius triplet (φ, λ, R).

The coordinates of a control point in a body-fixed frame can be given by

$$\mathbf{x} = \mathbf{R}(\cos\varphi\cos\lambda, \cos\varphi\sin\lambda, \sin\varphi)^T \qquad (1)$$

From (1).

$$\frac{\partial \mathbf{x}}{\partial \varphi} = R(-\sin\varphi\cos\lambda, -\sin\varphi\sin\lambda, \cos\varphi)^T$$
$$\frac{\partial \mathbf{x}}{\partial \lambda} = R(-\cos\varphi\sin\lambda, \cos\varphi\cos\lambda, 0)^T$$
$$\frac{\partial \mathbf{x}}{\partial R} = (\cos\varphi\cos\lambda, \cos\varphi\sin\lambda, \sin\varphi)^T$$

Define the right ascension and declination of the target body's north pole by

$$\alpha = \alpha_0 + \overline{\alpha}(t) \qquad (2)$$
$$\delta = \delta_0 + \overline{\delta}(t) \qquad (3)$$

where t is the interval between the J2000 epoch and the time the image was taken, α_0 and δ_0 are constants, and $\overline{\alpha}$ and $\overline{\delta}$ are known functions of t.

Define the spin-axis-inertial to Earth-equatorial-inertial transformation matrix as

$$\mathbf{M} = \begin{bmatrix} -\sin\alpha & -\cos\alpha\sin\delta & \cos\alpha\cos\delta \\ \cos\alpha & -\sin\alpha\sin\delta & \sin\alpha\cos\delta \\ 0 & \cos\delta & \sin\delta \end{bmatrix} \qquad (4)$$

[page 3]

From (2) and (4),

$$\frac{\partial \mathbf{M}}{\partial \alpha_0} = \begin{bmatrix} -\cos\alpha & \sin\alpha\sin\delta & -\sin\alpha\cos\delta \\ -\sin\alpha & -\cos\alpha\sin\delta & \cos\alpha\cos\delta \\ 0 & 0 & 0 \end{bmatrix}$$

From (3) and (4),

$$\frac{\partial \mathbf{M}}{\partial \delta_0} = \begin{bmatrix} 0 & -\cos\alpha\cos\delta & -\cos\alpha\sin\delta \\ 0 & -\sin\alpha\cos\delta & -\sin\alpha\sin\delta \\ 0 & -\sin\delta & \cos\delta \end{bmatrix}$$

Define the prime meridian of the target body by

$$w = w_0 + \dot{w}t + \overline{w}(t) \qquad (5)$$

where w_0 is a constant defining the zero meridian, $\dot{w}$ is the target body's rotation rate, and $\overline{w}$ is a known function of t.

From (5).

$$\frac{\partial w}{\partial \dot{w}} = t$$

Define the body-fixed to spin-axis-inertial transformation matrix by

$$\mathbf{V} = \begin{bmatrix} \cos w & -\sin w & 0 \\ \sin w & \cos w & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (6)$$

From (6),

$$\frac{\partial \mathbf{V}}{\partial \mathbf{w}} = \begin{bmatrix} -\sin w & -\cos w & 0 \\ \cos w & -\sin w & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{V}}{\partial \dot{w}} = \frac{\partial \mathbf{V}}{\partial w} \bullet \frac{\partial w}{\partial \dot{w}}$$

[page 4]

The coordinates of a control point in the J2000 Earth equatorial inertial system are given by

$$\hat{\mathbf{s}} = \mathbf{MVx} \qquad (7)$$

From (1)-(6),

$$\frac{\partial \hat{\mathbf{s}}}{\partial \mathrm{P}} = \mathbf{MV}\frac{\partial \mathbf{x}}{\partial \mathrm{P}}, \mathrm{P} \in \{\varphi, \lambda, R\}$$
$$\frac{\partial \hat{\mathbf{s}}}{\partial \mathrm{P}} = \frac{\partial \mathbf{M}}{\partial \mathrm{P}}\mathbf{Vx}, \mathrm{P} \in \{\alpha_0, \delta_0\}$$
$$\frac{\partial \hat{\mathbf{s}}}{\partial \dot{w}} = \mathbf{M}\frac{\partial \mathbf{V}}{\partial \dot{w}}\mathbf{x}$$

Define **s** as the spacecraft position vector in J2000 coordinates. The range vector, i.e., the vector subtended between the spacecraft and the control point, is then given by

$$\mathbf{s}^{\mathrm{v}} = \hat{\mathbf{s}} - \mathbf{s} \qquad (8)$$

Then it is clear that

$$\frac{\partial \mathbf{s}^{\mathrm{v}}}{\partial \mathrm{P}} = \frac{\partial \hat{\mathbf{s}}}{\partial \mathrm{P}}, \mathrm{P} \in \{\varphi, \lambda, R, \alpha_0, \delta_0, \dot{w}\}$$

Define the camera orientation matrix, which transforms Earth equatorial inertial coordinates to camera coordinates, by

$$\mathbf{C} = \begin{bmatrix} -\sin\hat{\alpha}\cos\kappa - \cos\hat{\alpha}\sin\hat{\delta}\sin\kappa & \sin\hat{\alpha}\cos\kappa - \sin\hat{\alpha}\sin\hat{\delta}\sin\kappa & \cos\hat{\delta}\sin\kappa \\ \sin\hat{\alpha}\sin\kappa - \cos\hat{\alpha}\sin\hat{\delta}\cos\kappa & -\cos\hat{\alpha}\sin\kappa - \sin\hat{\alpha}\sin\hat{\delta}\cos\kappa & \cos\hat{\delta}\cos\kappa \\ \cos\hat{\alpha}\cos\hat{\delta} & \sin\hat{\alpha}\cos\hat{\delta} & \sin\hat{\delta} \end{bmatrix}, \qquad (9)$$

where $\hat{\alpha}$ and $\hat{\delta}$ define the direction of the optical axis, and κ specifies the picture rotation angle in the camera focal plane system.

From (9),

[page 5]

$$\frac{\partial \mathbf{C}}{\partial \hat{\alpha}} = \begin{bmatrix} -\cos\hat{\alpha}\cos\kappa + \sin\hat{\alpha}\sin\hat{\delta}\sin\kappa & -\sin\hat{\alpha}\cos\kappa - \cos\hat{\alpha}\sin\hat{\delta}\sin\kappa & 0 \\ \cos\hat{\alpha}\sin\kappa + \sin\hat{\alpha}\sin\hat{\delta}\cos\kappa & \sin\hat{\alpha}\sin\kappa - \cos\hat{\alpha}\sin\hat{\delta}\cos\kappa & 0 \\ -\sin\hat{\alpha}\cos\hat{\delta} & \cos\hat{\alpha}\cos\hat{\delta} & 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{C}}{\partial \hat{\delta}} = \begin{bmatrix} -\cos\hat{\alpha}\cos\hat{\delta}\sin\kappa & -\sin\hat{\alpha}\cos\hat{\delta}\sin\kappa & -\sin\hat{\delta}\sin\kappa \\ -\cos\hat{\alpha}\cos\hat{\delta}\cos\kappa & -\sin\hat{\alpha}\cos\hat{\delta}\cos\kappa & -\sin\hat{\delta}\cos\kappa \\ -\cos\hat{\alpha}\sin\hat{\delta} & -\sin\hat{\alpha}\sin\hat{\delta} & \cos\hat{\delta} \end{bmatrix}$$

$$\frac{\partial \mathbf{C}}{\partial \kappa} = \begin{bmatrix} \sin\hat{\alpha}\sin\kappa - \cos\hat{\alpha}\sin\hat{\delta}\cos\kappa & -\cos\hat{\alpha}\sin\kappa - \sin\hat{\alpha}\sin\hat{\delta}\cos\kappa & \cos\hat{\delta}\cos\kappa \\ \sin\hat{\alpha}\cos\kappa + \cos\hat{\alpha}\sin\hat{\delta}\sin\kappa & -\cos\hat{\alpha}\cos\kappa + \sin\hat{\alpha}\sin\hat{\delta}\sin\kappa & -\cos\hat{\delta}\sin\kappa \\ 0 & 0 & 0 \end{bmatrix}$$

Define $\boldsymbol{\gamma} = (\xi, \eta, \varsigma)^T$ as the range vector in camera coordinates,

$$\boldsymbol{\gamma} = \mathbf{C}\mathbf{s}^{v} \qquad (10)$$

By (8) and (10),

$$\frac{\partial \gamma}{\partial \mathrm{P}} = \mathbf{C}\frac{\partial \mathbf{s}^v}{\partial \mathrm{P}}, \mathrm{P} \in \{\varphi, \lambda, R, \alpha_0, \delta_0, \dot{w}\}$$
$$\frac{\partial \gamma}{\partial \mathrm{P}} = \frac{\partial \mathbf{C}}{\partial \mathrm{P}}\mathbf{s}^v, \mathrm{P} \in \{\hat{\alpha}, \hat{\delta}, \kappa\}$$

Define the coordinates of a control point in the camera focal plane by

$$x_c = \frac{\xi f}{\varsigma}, y_c = \frac{\eta f}{\varsigma} \qquad (11)$$

where f is the focal length of the camera. From (11),

$$\left.\begin{aligned}\frac{\partial x_c}{\partial \mathrm{P}} &= \frac{f}{\varsigma^2}\left(\varsigma\frac{\partial \xi}{\partial \mathrm{P}} - \xi\frac{\partial \varsigma}{\partial \mathrm{P}}\right)\\ \frac{\partial y_c}{\partial \mathrm{P}} &= \frac{f}{\varsigma^2}\left(\varsigma\frac{\partial \eta}{\partial \mathrm{P}} - \eta\frac{\partial \varsigma}{\partial \mathrm{P}}\right)\end{aligned}\right\}, \mathrm{P} \in \{\varphi, \lambda, R, \alpha_0, \delta_0, \dot{w}, \hat{\alpha}, \hat{\delta}, \kappa\}$$

[page 6]

Suppose there are a total of n control points, p pictures, m measurements and k = 3n + 3p + 3 unknowns.

Define by $\mathbf{e_i}$ the coordinate axis vectors in $\Re^k$, i.e.,

$$\mathbf{e_1} = (1,0,\cdots,0)^T, \cdots, \mathbf{e_k} = (0,\cdots,0,1)^T$$

and define

$$\mathbf{u} = \left(x_c^{(1)}, y_c^{(1)}, \cdots, x_c^{(m)}, y_c^{(m)}\right).$$

Then the observation equations may be written in the form

$$\mathbf{Lz} = \mathbf{f} \qquad (12)$$

where

$$\mathbf{Le_1} = \frac{\partial \mathbf{u}}{\partial \varphi_1}, \mathbf{Le}_2 = \frac{\partial \mathbf{u}}{\partial \lambda_1}, \mathbf{Le}_3 = \frac{\partial \mathbf{u}}{\partial R_1}$$

…

$$\mathbf{Le}_{3n-2} = \frac{\partial \mathbf{u}}{\partial \varphi_n}, \mathbf{Le}_{3n-1} = \frac{\partial \mathbf{u}}{\partial \lambda_n}, \mathbf{Le}_{3n} = \frac{\partial \mathbf{u}}{\partial R_n}$$

$$\mathbf{Le}_{3n+1} = \frac{\partial \mathbf{u}}{\partial \hat{\alpha}_1}, \mathbf{Le}_{3n+2} = \frac{\partial \mathbf{u}}{\partial \hat{\delta}_1}, \mathbf{Le}_{3n+3} = \frac{\partial \mathbf{u}}{\partial \kappa_1}$$

…

$$\mathbf{Le}_{3n+3p-2} = \frac{\partial \mathbf{u}}{\partial \hat{\alpha}_p}, \mathbf{Le}_{3n+3p-1} = \frac{\partial \mathbf{u}}{\partial \hat{\delta}_p}, \mathbf{Le}_{3n+3p} = \frac{\partial \mathbf{u}}{\partial \kappa_p}$$

$$\mathbf{Le}_{3n+3p+1} = \frac{\partial \mathbf{u}}{\partial \alpha_0}, \mathbf{Le}_{3n+3p+2} = \frac{\partial \mathbf{u}}{\partial \delta_0}, \mathbf{Le}_{3n+3p+3} = \frac{\partial \mathbf{u}}{\partial \dot{w}}$$

$$\mathbf{z} = \left(\Delta\varphi_1, \Delta\lambda_1, \Delta R_1, \cdots, \Delta\varphi_n, \Delta\lambda_n, \Delta R_n, \Delta\hat{\alpha}_1, \Delta\hat{\delta}_1, \Delta\kappa_1, \cdots, \Delta\hat{\alpha}_p, \Delta\hat{\delta}_p, \Delta\kappa_p, \Delta\alpha_0, \Delta\delta_0, \Delta\dot{w}\right)$$

[page 7]

If one does not wish to solve for some of the variables, then the observation equations degenerate in an obvious manner. Note also that a special case of the above equations allows for the solution of a single body-wide radius rather than a radius at each control point.

The normal equations are formed by multiplying both sides of (12) by $\mathbf{L}^T$ to get

$$\left(\mathbf{L}^T\mathbf{L}\right)\mathbf{z} = \mathbf{L}^T\mathbf{f} \qquad (13)$$

Solving (13) gives the desired corrections, viz.,

$$\left.\begin{aligned} \varphi_j^{(i+1)} &= \varphi_j^{(i)} + \Delta\varphi_j^{(i)} \\ \lambda_j^{(i+1)} &= \gamma_j^{(i)} + \Delta\gamma_j^{(i)} \\ R_j^{(i+1)} &= R_j^{(i)} + \Delta R_j^{(i)} \end{aligned}\right\}, j = 1,2,\cdots,n$$

$$\left.\begin{aligned} \hat{\alpha}_j^{(i+1)} &= \hat{\alpha}_j^{(i)} + \Delta\hat{\alpha}_j^{(i)} \\ \hat{\delta}_j^{(i+1)} &= \hat{\delta}_j^{(i)} + \Delta\hat{\delta}_j^{(i)} \\ \kappa_j^{(i+1)} &= \kappa_j^{(i)} + \Delta\kappa_j^{(i)} \end{aligned}\right\}, j = 1,2,\cdots,p$$

$$\left.\begin{aligned}\alpha_0^{(i+1)} &= \alpha_0^{(i)} + \Delta\alpha_0^{(i)} \\ \delta_0^{(i+1)} &= \delta_0^{(i)} + \Delta\delta_0^{(i)} \\ \dot{w}^{(i+1)} &= \dot{w}^{(i)} + \Delta\dot{w}^{(i)}\end{aligned}\right\}, j = 1,2,\cdots, p$$

Iteration is continued until the variables no longer change.

[page 8]

3. SOFTWARE DESCRIPTION

[…transcription to be continued…]

Appendix II

HiRISE Sensor Model Information

E-mail from R. Kirk to HiGeom group mailing list of 2004 November 22

= = = = = = = = = =

"Randolph L Kirk" <rkirk@usgs.gov>
Sent by: owner-unix@flagmail.wr.usgs.gov
11/22/2004 05:49 PM
Please respond to higeom

To: HiGeom Group <higeom@flagmail.wr.usgs.gov>
cc:
Subject: [higeom] Sensor model formulation for ISIS 3

Jim:
I apologize... I evidently never wrote anything down about sensor model approaches after our last meeting on the subject. I'm happy to make a concrete suggestion how to proceed. What I plan to suggest is not the only workable approach, but it is one of a couple that are general enough to make me happy about our prospects of being able to handle all our sensors in one framework.

I'll start by defining the problem. The overall problem of computing ground coordinates from image coordinates consists of the following steps:

1.0) Given sample and line coordinates in an image, determine the sample and line in the physical detector, taking account of pixel summation.
1.1) Given the physical detector sample/line coordinates, determine the XY coordinates of the pixel in the detector plane. These are augmented with a Z coordinate that is closely related to the focal length, or at least the distance from the detector plane to the principal point.
1.2) Use a radial distortion polynomial to compute new plane coordinates X'Y' from XY.
1.3) Normalize the vector X'Y'Z appropriately, to get a unit vector pointing from the camera to the feature, expressed in the coordinate system of the camera.
1.4) Rotate this vector into appropriate (body-fixed) coordinates.
1.5) Find the intersection of the line through the spacecraft position with indicated direction (the unit vector) with the target. This can be an iterative process if one is using a DTM instead of an ellipse to describe the target.

The problem of computing image coordinates from ground coordinates consists of the following steps:

2.1) Take the difference between the feature coordinates (body fixed) and spacecraft coordinates to get the vector from feature to spacecraft.
2.2) Normalize this to a unit vector
2.3) Rotate this vector from body-fixed coordinates to camera coordinates
2.4) De-normalize the unit vector into X'Y'Z where Z is related to the focal length.
2.5) Apply the optical distortion in the opposite direction to above, to get XY from X'Y'.
2.6a) For a frame sensor, calculate line and sample from X and Y
2.6b) For a pushbroom sensor, calculate perpendicular distance from XY to the row of detector pixels. Iterate 2.1-2.6 adjusting the assumed time of imaging until this distance is zero. Calculate sample from XY and line from time.

What we are concerned with here are strictly the steps 1.1 and 2.6 (and we also don't care about the details of how we iterate to adjust the time in 2.6b, but we do care about calculating the perpendicular distance).

The vector we are going to compute to start with is the vector from a detector pixel to the optics principal point and on toward the target. The detectors are behind the optics in this conventional view, and they are also upside down. It's equivalent, conventional, and easier to pretend that the detector plane is rightside up and in front of the principal point. Then a line from the principal point to the pixel goes on through it and out into the world to intersect the target. I can't draw this but can supply a figure. Anyway, if the nominal optical plane is the XY plane (Z=0) and Z increases away from the target (out the back of the camera) then the principal point would normally be located at 0,0,f where f is usually called the focal length.
(If the camera isn't actually focused at infinity, the distance along the Z axis won't be the focal length, but we can ignore that distinction.)
The vector from principal point to pixel is then X,Y,-f.
In the past, we've gotten X and Y by assuming that a particular pixel (the boresight, at sb, lb) is located at 0,0 and that the axes are along the rows and columns, so (say) X = delta * (s - sb) and Y = delta * (lb-l) where delta is the pixel pitch. The sb and lb might be hard coded for a given instrument; for a scanner we might even assume Y = 0 or Y = constant instead of the second equation. We might have a detector with different pixel pitches in line and sample but we'd normally fix the image by resampling instead of dealing with this. Then we can store f/delta (the "focal length in pixels") in the code instead of f and ignore delta in the above equations. We'd like to generalize on a lot of this stuff so we don't have to use a separate set of assumptions for each detector in the future.

We also get a bunch of new complications when we have cameras with multiple detectors. We could, perhaps, implement every detector as if it were a completely separate camera, but if we make a more physical model with one set of axes and one "focal length" (principal point

coordinate relative to the Z=0 plane) we will have to allow some of the detectors to be rotated and/or translated relative to that coordinate system. These rotations/translations are not limited to the XY plane; the detector could be above/below the plane (making its apparent image scale different from the others) or it could be tipped, making the image scale vary across the field of view.

One way to model these effects would be to use a general 3D transformation between camera coordinates and detector coordinates. This could be implemented with NAIF Frames but it does not have to be. We would then have a simple relation like

```
Xdet =    delta * s
Ydet = - delta * L
Zdet = 0
```

in the detector system, and a rotation/translation between this and the camera system to get X,Y,Z. Then we would have to form our vector as X,Y,Z-f instead of just X,Y,Z. I've written L instead of l, but for a frame camera this would indeed just be the line number. For a scanner, the true image line l would only appear by determining the time, so L would either be 0 or we could use it as a bias factor to account for the effective location of different TDI buffer ranges being in different places in the coordinate system.

The approach just outlined models the real physics but it has a couple of drawbacks. We have to make a bunch of NAIF calls, which do a bunch of matrix and vector operations, every time we want to project a pixel. Then we have to form Z-f rather than having a fixed z-component to the vector of interest. This is going to be somewhat slow.

Also, when we go from ground to image for a scanner, we need to determine the distance from the feature point, projected into the detector plane, to the detector array. But with the full 3D rotation the detector array doesn’t lie in the plane. So some cleverness would be needed to figure out what distance needs to be reported. The answer is that it's the 2D distance between the point projected into the detector plane and the detector line projected into the detector plane.

The second approach to describing the detectors and carrying out the calculations is based on precisely the projection into the nominal detector plane that we had to do to get the distance anyhow, but it uses the projected coordinates for everything. The result is a set of 2D matrix/vector operations that will be faster than the 3D operations in the other approach. The matrix coefficients will not be the physical transformation coefficients between frames, which might be seen as a disadvantage, but they can be related to real physical quantities. Also, we can "own" these quantities ourselves instead of asking the NAIF frames software to manage them, so this will also give a speed advantage.

The relation between sample and "line" L (in the sense above, either the detector line or some kind of mode-related offset) and XY in the

case where the detector does happen to lie right in the nominal plane is a 2D rotation and translation. It takes the form
X = C10 + C11 * s + C12 * L
Y = C20 + C21 * s + C2 *2 L

with certain restrictions on C11, C12, C21, C22 such that they form a rotation matrix times a scale factor. Furthermore we know the scale factor is delta if the detector lies in the plane. The restrictions are C11*C21+C12*C22=0, C11*C12+C21*C22=0 and C11*C22-C21*C12=delta*delta. It's actually easier to start from the rotation and make C11 = C22 = delta*cos(theta) and C12 = -C21 = delta*sin(theta) to build such a matrix.

The first generalization we can make to this model is to let the scale be other than delta. This corresponds to letting the detector lie above or below the nominal detector plane but parallel to it. We can relate the coefficients to physical quantities by forming the effective scale C11*C22-C12*C21 and comparing it to the true scale delta. They will be in the ratio of f to f-Z.

Relaxing the other two constraints lets the scale be different in the two directions and lets the line and sample directions be not perpendicular. In both cases, it could be either the physical detector that has these properties (e.g., nonsquare pixels) or it could be an effect of projection into the nominal plane.

However, if the detector isn't parallel to the nominal detector plane, the distance of different pixels above/below the plane will be different and the effective scale will actually change across the image. The physical way to model this is to let Z be a linear function of pixel coordinates and then make the scale proportional to f/(f-Z). For small departures from the ideal model, i.e., Z << f, we get f/(f-Z) = 1(1-Z/f) approximately equal to 1+Z/f and the scale will vary linearly with pixel coordinates. If the scale varies linearly, then the projected coordinate will vary quadratically.
Thus, the general expression will become

X = C10 + C11 * s + C12 * L + C13 * s^2 + C14 * s * L + C15 * L^2
Y = C20 + C21 * s + C22 * L + C23 * s^2 + C24 * s * L + C25 * L^2

Once again, this would be subject to certain restrictions. The general form is capable of representing a detector whose rows and columns actually curve in a parabolic shape, whereas all we want is to have them project onto the nominal plane with a varying scale. We can get the necessary constraints with not that much more work than I did in the case above, but I'm not sure it's worth it.

What I recommend is that we

a) Implement the code in the form just given, with 12 coefficients.
b) Make initial estimates of the values for the coefficients for a given detector based on our understanding of the idealized geometry of the camera. This means setting the quadratic coefficients to zero,

calculating C11 = C22 = delta, C12 = -C21 = 0, and figuring out the nominal offset.
c) Adjust the coefficients to fit the calibration data. This would be done so as not to introduce the full complexity that is possible.

For example, we would first introduce the rotation theta used above. Then we might (for a line scanner) introduce a scale difference that depends only on sample, on the presumption that the variations in L are small anyway. If we do this in such a way as to keep the projection of the detector into the nominal plane a straight line, we would have

C13 = eps * C11
C24 = eps * C21
C14=C15=C23=C25=0

where eps is yet another arbitrary constant. We could actually set up the model with this restriction built into it, i.e.,

X = C10 + C11 * s*(1+eps*s) + C12 * L
Y = C20 + C21 * s*(1+eps*s) + C22 * L

but then we'd probably want to do something different for array detectors. My opinion is that it's best to set up the quadratic mapping of detector line and sample to X and Y as given above, and then use restraint when we go to set up the coefficients for different cameras. For area detectors we'd presumably always set C13=C14=C15=C23=C24=C25=0, but maybe we would someday encounter a camera with multiple area CCDs that aren't co-planar, and we'd be happy we had the more general format. Or we might encounter a pushbroom camera whose line array got screwed down a little too hard so it's actually curved, and we could use the general coefficients to model this.

The remaining problem is how to solve for the distance from a trial point projected into the plane to the detector curve when the curve is no longer a straight line.

In the restricted case I'd plan to start with, the curve IS still a straight line and we could just define T = s*(1+eps*s) and have a line in XY parameterized by the variable T and the fixed quantity L. Then the distance from a point at Xpoint,Ypoint to this line is easy to compute. I don't have the formula memorized but it's a standard result of analytic geometry: easy to derive, even easier to look up.

The perpendicular distance from a point to a quadratic curve should be computable fairly easily but it could be multiple valued in the general case. I am going to break off now and go home before the roads freeze, so I'll work on this part of the problem at another time.

Cheers,
Randy

Dr. Randolph L. Kirk
Astrogeology Team Ph. 928-556-7020

U.S. Geological Survey Fax 928-556-7014
2255 N. Gemini Dr.
Flagstaff, AZ 86001 USA
NOTE new email address: ASTROG::RKIRK and
rkirk@flagmail.wr.usgs.gov are now rkirk@usgs.gov

[Minor typographical errors have been corrected in the above.]