A shortened version of this paper appears in iSAIRAS'99.

Precise Image-Based Motion Estimation for Autonomous Small Body Exploration

Andrew E. Johnson and Larry H. Matthies

Jet Propulsion Laboratory
Mail Stop 125-209, 4800 Oak Grove Drive
Pasadena, CA 91109
(aej,lhm}@robotics.jpl.nasa.gov

November 1998

Abstract

Comets and asteroids play a prominent role in NASA's roadmap for solar system exploration because they hold answers to questions about the origin of the solar system. NASA is planning multiple small body missions that range in scope from near body flybys to complete sample return [4][8][12][15]. This paper presents an algorithm for autonomous onboard motion estimation that will enable the precision guidance and landing necessary for small body sample return. Our techniques are based on automatic feature tracking between a pair of descent camera images followed by two frame motion estimation and scale recovery using laser altimetry data. The output of our algorithm is an estimate of rigid motion (attitude and position) and motion covariance between frames. This motion estimate can be passed directly to the spacecraft guidance navigation and control system to enable rapid execution of safe and precise trajectories.

1 Introduction

Due to the small size, irregular shape and variable surface properties of small bodies (see Figure 2), accurate motion estimation is needed for safe and precise small body exploration. Because of the communication delay induced by the large distances between the earth and targeted small bodies, landing on small bodies must be done autonomously using on-board sensors and algorithms. Current navigation technology does not provide the precision necessary to accurately land on a small bodies, so novel motion estimation techniques must be developed. Computer vision offers a possible solution to precise motion estimation.

Historically, optical navigation has been used for orbit determination and instrument pointing during close fly-bys of small bodies and moons of the outer planets. Generally, this has been implemented by ground-based image processing to extract centroids of small reference targets like asteroids and moons from which target relative spacecraft attitude and position are computed.

The Near Earth Asteroid Rendezvous (NEAR), a current mission that will rendezvous with asteroid Eros 433 in February 2000, uses optical navigation extensively for orbit determination and small body 3-D modeling [12]. Their base-lined navigation technique will combine manually designated landmarks from imagery of Eros and radiometric data to compute and control the trajectory of the orbiter. Simulations show that after a week of ground-based processing, the orbit of the NEAR spacecraft can be determined to 100's of meters from an orbit of 500 kilometers. Without optical navigation, the accuracy of the orbit determination from radiometric data would be closer to 5 kilometers. The NEAR mission will clearly demonstrate the effectiveness of optical navigation. However, this ground-based paradigm will not map to missions involving small body exploration and landing.

Small body exploration requires multiple precise target relative maneuvers during a brief descent to the surface. The round trip light time prohibits the determination of the necessary trajectory control maneuvers on the ground. Furthermore, typical onboard position sensors do not have the accuracy needed for small body landing (e.g., during a small body descent taking a few hours accelerometer errors will grow to the kilometer level). However, the required

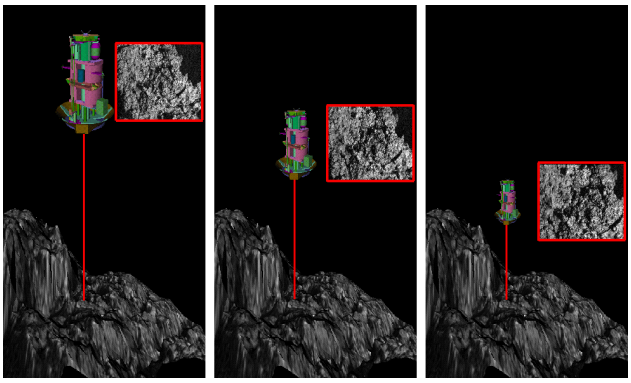


Figure 1: Image-based precision landing. As the spacecraft descends to the surface, images and laser altimetry are processed to determine the motion of the spacecraft.

The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract from the National Aeronautics and Space Administration.

positional accuracies can be obtained during small body landing if autonomous real-time optical navigation methods are developed.

The Deep Space 1 mission as part of the New Millennium Program is flying an autonomous optical navigation technology demonstration. The DS-1 AutoOpNav system will use onboard centroiding of reference asteroids for autonomous navigation during small body fly-bys [2]. They expect to obtain automatic position estimates with accuracies on order of 100 kilometers. For scientific instrument pointing purposes, this accuracy is sufficient. Controlled small body landing will require much better position and motion estimation accuracies. Furthermore, since the appearance of the small body is variable, small body landing cannot always rely on reference landmarks for navigation. The DS-1 AutoOpNav system will demonstrate autonomy and computer vision in space, however for small body landing a more versatile and accurate system is required.

This paper describes a fully autonomous and onboard solution for accurate and robust motion estimation near a proximal small body. Our techniques are based on automatic feature tracking between a pair of images followed by two frame motion estimation and scale recovery using laser altimetry data. The output of our algorithm is an estimate of rigid motion (attitude and position) and motion covariance between frames. This motion estimate can be passed directly to the spacecraft guidance navigation and control system to enable rapid execution of safe and precise trajectories.

The rest of the paper is organized as follows. In Section 2 we describe in detail our complete motion estimation algorithm. In Section 3 we present tests on real data which show motion estimation rates of 4 Hz with positional accuracies of 4.5% of the distance traveled and attitudinal accuracies of 0.06 degrees. In Section 4 we describe a set of tests which are used to predict the performance of the algorithms. Using this simulation, we show positional accuracies of 0.22 m when descending 65 m from an altitude of 1000 m, horizontal landing position accuracies of 3.6 m when descending from 1000 m, and rotational accuracies of 0.006˚ when off axis pointing of 0.6˚ are possible. Finally in Section 5 we present conclusions.

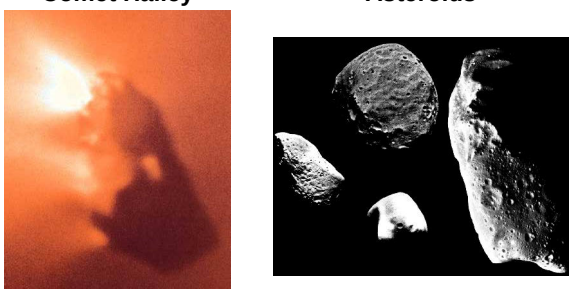


Figure 2: Small bodies.

2 Motion Estimation

Motion estimation from images has a long history in the machine vision literature. The algorithm presented in this paper falls in the category of two-frame feature-based motion estimation algorithms. To obtain complete 6 DOF motion estimates, our algorithm is augmented by altimeter measurements for scale estimation.

Once the spacecraft sensors are pointed at the small body surface, our algorithm works as follows (see Figure 1 for a pictorial description). At one time instant a descent camera image and a laser altimeter reading are taken. A short time later, another image and altimeter reading are taken. Our algorithm then processes these pairs of measurements to estimate the rigid motion between readings. There are multiple steps in our algorithm. First, distinct features, which are pixels that can be tracked well across multiple images, are detected in the first image. Next, these features are located in the second image by feature tracking. Given these feature matches, the motion state and covariance of the spacecraft, up to a scale on translation, are computed using a two stage motion estimation algorithm. Finally the scale of translation is computed by combining altimetry with the motion estimates using one of two methods which depend on the descent angle. The block diagram for motion estimation is shown in Figure 3.

2.1 Feature Detection

The first step in two-frame motion estimation is the extraction of features from the first image. Features are pixel locations and the surrounding image intensity neighborhood (call this an image window) that can be tracked well across multiple images that may under go arbitrary, but small, changes in illumination or viewing direction. A qualitative definition of a good feature is an image window that has strong texture variations in all directions.

Feature detection has been studied extensively and multiple proven feature detection methods exist. Consequently, we elected to implement a proven feature detection method instead of redesigning our own. Since processing speed is a very important design constraint for our application, we selected the state of the art feature detection algorithm of Benedetti and Perona [2]. This algorithm is an implementation of the well know Shi-Tomasi feature detector and tracker [16] modified to eliminate transcendental arithmetic. Although they ultimately implemented their algorithm in hardware on a reconfigurable computer, their algorithmic speed enhancements also decrease the running time of software implementations.

2.1.1. Shi-Tomasi-Kanade Feature Detection

The theoretical derivation of their feature detector is explained fully in [2]; we will detail our software implementation of this algorithm so that the necessary computations are apparent. Let *I(p,q)* be function defining image intensity for an image. We would like to determine if an image window containing *M* pixels centered on pixel $\boldsymbol{p} = (p,q)$ is distinctive enough to be considered a feature. First form the matrix

$$G = \begin{bmatrix} \sum_{k=1}^{M} (I_p^k)^2 & \sum_{k=1}^{M} I_p^k I_q^k \\ \sum_{k=1}^{M} I_p^k I_q^k & \sum_{k=1}^{M} (I_q^k)^2 \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \quad (1)$$

from the partial derivatives of image intensity I_p and I_q computed using finite differences

$$I_p(\boldsymbol{p}) = \frac{I(p+1,q) - I(p-1,q)}{2}$$
$$I_q(\boldsymbol{p}) = \frac{I(p,q+1) - I(p,q-1)}{2} \quad (2)$$

The criterion for the image pixel to be a feature is that the two eigenvalues λ_1 and λ_2 of *G* be greater than a threshold λ_t (i.e., $\lambda_2 > \lambda_1 > \lambda_t$.) As shown in [2], this requirement is the same as the following

$$(a - \lambda_t)(c - \lambda_t) > 0 \qquad a > \lambda_t \quad (3)$$

Surfaces of celestial bodies generally appear highly textured[ref], so good features to track are expected to be plentiful. Usually feature detection algorithms exhaustively search the image for every distinct feature. However, when the goal is motion estimation, only a relatively small number of features need to be tracked (~100). Consequently, we can speed up feature tracking by using a random search strategy instead of exhaustive search while still guaranteeing that the required number of features are detected. Suppose that *N* features are needed for motion estimation. Our detection algorithm selects a pixel at random from the image (uniform distribution in row and column directions). It then computes the image derivatives (unless they have already been computed from a previous feature detection) and the *G* matrix for a neighborhood of *M* pixels around the pixel. Next, the test in Equation 3 is applied to the pixel. If the pixel passes the test, it becomes a detected feature. This procedure is repeated until *N* features are detected.

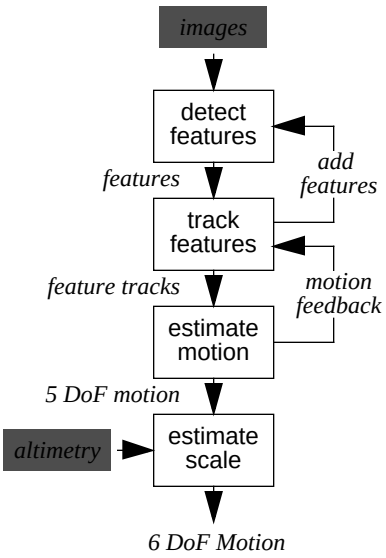


Figure 3: Block diagram for motion estimation.

2.1.2. Processing Issues

The running time of the randomized detection algorithm depends on the number of features required, and the number of pixels in the scene which pass the feature detection threshold. In general we have found that the randomized search algorithm increases the running time of feature detection by an order of magnitude over traditional exhaustive search. In Figure 4 a comparison of the best 50 features selected in an 640x480 image (top) using exhaustive search versus 50 features selected using our randomized algorithm (bottom) are shown. The exhaustive search algorithm took 11.19 seconds while the randomized algorithm took 0.29 seconds to compute on an 174 Mhz R10000 SGI O^2. As the figure shows, the features selected by the random search algorithm occur in similar image locations as those features selected using exhaustive search indicating that random search is detecting appropriate features in highly textured areas of the image.

The majority of the memory required for feature tracking come from the 2-D character array for storing the image (assuming an 8 bit image) and two 2-D integer arrays for the image derivatives. For a 1024x1024 imager the memory requirement is 9 MB of RAM. At the expense of increased processing time, the image derivatives can be computed from the image for each pixel in every window investigated for features. This will eliminate the need for the image derivative arrays and will reduce the memory requirements to 1 MB.

There are two parameters in feature detection: the number of pixels *M* comprising the window in which the *G* matrix is computed; and the threshold on the eigenvalue λ_τ, above which a pixel is considered a feature. For convenience, the window around a pixel is square. In Figure 4, the window is 5x5, so *M*=25. In general, the size of the window will dictate the scale of the features detected. Small windows will detect precisely localized small scale features, but they are sensitive to image noise. Larger windows will detect large scale features with less location accuracy, but will be less susceptible to noise. Since we are

using feature detection for motion estimation, the precise localization of features as well as rapid processing time is very important. Therefore when selecting a value for M, we attempt to make it as small as possible while still obtaining accurate feature detection and tracking. Automatic methods can be developed, but currently the user sets M based on feature tracking performance.

Since λ_τ is a threshold on the eigenvalues of matrix made from the M additions of pixel derivative products, it will be proportional to M. To remove this dependence, we scale λ_τ by M which makes λ_τ depend only on the texturedness in the image. Since the texturedness of the scene does not vary greatly for precision landing, λ_τ will only have to be set once for each small body. For the features in Figure 4, λ_τ was set to 2500. Automatic methods requiring little additional computation can be developed for setting λ_τ, but currently the user sets it based on feature detection performance.

2.2 Feature Tracking

The next step in motion estimation is to locate the features detected in the first frame in the second frame. This procedure is called feature tracking. As with feature detection, there exist multiple methods for feature tracking in the machine vision literature. Feature tracking can be split in to two groups of algorithms: correlation based methods[11] and optical flow based methods [16]. Correlation based methods are appropriate when the motion of features in the image is expected to be large. For small motions, optical flow based methods are more appropriate because in general they require less computation than correlation methods[1]. We have chosen an optical flow based method for feature tracking because in our application of precision landing, we know a-priori that the motion between image frames will be small. Furthermore, our selected method of feature detection is derived from optical flow based feature tracking; the features that are detected are exactly the features that give the best results for feature tracking based on optical flow.

We use the Shi-Tomasi-Kanade feature tracker which seeks to minimize the intensity difference ε between two images I and J

$$\varepsilon = \sum_{k=1}^{M} (J^k(\boldsymbol{p}+\boldsymbol{d}) - I^k(\boldsymbol{p}))^2 \tag{4}$$

over the space of possible feature image translations $\boldsymbol{d}$ in the vicinity of the feature at pixel location $\boldsymbol{p}$ in image I. To minimize Equation 4, it is differentiated with respect to $\boldsymbol{d}$ and the result is set to zero. After linearizing the resulting system by truncating its Taylor expansion, the system of equations

$$G\boldsymbol{d} = \boldsymbol{e} \tag{5}$$

with G given by Equation 1 and $\boldsymbol{e}$ given by

$$\boldsymbol{e} = \begin{bmatrix} \sum_{k=1}^{M} (I^k(\boldsymbol{p}) - J^k(\boldsymbol{p}))I_p^k(\boldsymbol{p}) \\ \sum_{k=1}^{M} (I^k(\boldsymbol{p}) - J^k(\boldsymbol{p}))I_q^k(\boldsymbol{p}) \end{bmatrix} \tag{6}$$

can be solved for the feature motion $\boldsymbol{d}$.

Because of the linearization, the solution to Equation 5 does not minimize Equation 4 exactly. However using Equation 5, a Newton-Rhapson style iterative minimization can be used to solve for the feature motion exactly. The procedure is to first solve Equation 5 for $\boldsymbol{d_0}$ (G and $\boldsymbol{e}$ are constructed assuming $\boldsymbol{d} = \boldsymbol{0}$). Then iteratively solve Equation 5 for $\boldsymbol{d_i}$ with $\boldsymbol{e}$ replaced by

$$\boldsymbol{e_i} = \begin{bmatrix} \sum_{k=1}^{M} (I^k(\boldsymbol{p}) - J^k(\boldsymbol{p}+\boldsymbol{d_{i-1}}))I_p^k(\boldsymbol{p}) \\ \sum_{k=1}^{M} (I^k(\boldsymbol{p}) - J^k(\boldsymbol{p}+\boldsymbol{d_{i-1}}))I_q^k(\boldsymbol{p}) \end{bmatrix} \tag{7}$$

until $\boldsymbol{d_i}$ changes very little. Since $\boldsymbol{d_i}$ is a floating point value,

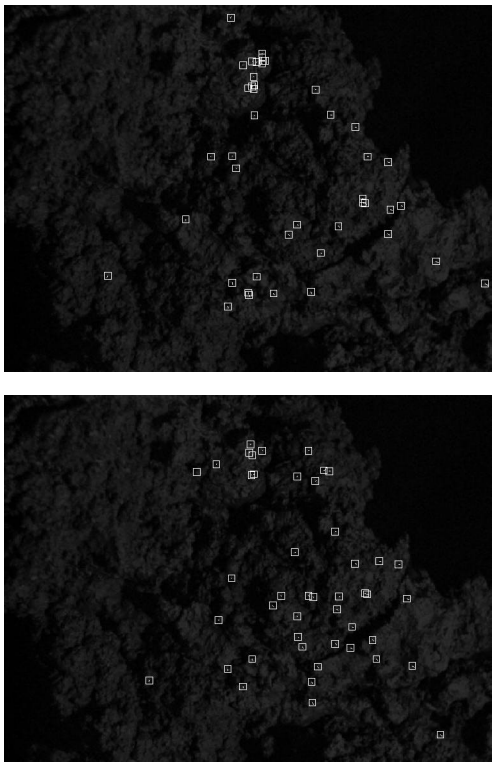


Figure 4: Detected features. Exhaustive search (top) and random search (bottom).

1. Correlation methods require $O(M^2)$ multiplications per feature while optical flow methods require $O(SM)$ multiplications per feature where S is the number of steps required to reach the local minimum of the SSD intensity surface. Intuitively, S will be on order $M^{1/2}$ because it is related to the distance between the pixel and the intensity local minimum. This results in $O(M^{3/2})$ multiplications per feature for optical flow based feature tracking.

constructing $\boldsymbol{e_i}$ requires bilinear interpolation of image *J*. For example, if we would like to determine the intensity of image *J* at floating point image coordinates $\boldsymbol{p} = [p,q]^T$

$$J(p, v) = \begin{array}{l}(1 - \underline{p})(1 - \underline{q})J(\underline{p}, \underline{q}) + \underline{p}(1 - \underline{q})J(\underline{p} + 1, \underline{q}) + \\ (1 - \underline{p})\underline{q}J(\underline{p}, \underline{q} + 1) + \underline{p}\underline{q}J(\underline{p} + 1, \underline{q} + 1)\end{array} \quad (8)$$

where $\underline{x}$ represents the floor of x and $\tilde{x} = x - \underline{x}$.

2.2.1. Processing Issues

The running time of feature tracking is *O(MN)* where *N* is the number of features being tracked and *M* is the size of the window used for tracking. The size of the window used in feature tracking does not have to be the same as the size of the window used in feature detection. In fact, for robust feature tracking, the window should be large enough that it contains the location of the pixel of the feature in the second frame. With large feature displacements between images, it may be necessary to increase the size of the window used for tracking to a value greater than that used for feature detection. For the feature tracks shown in Figure 5, the size of the window is 7x7 and the median displacement between features is 2.35 pixels; only 20 of 50 tracked features are shown for clarity. Tracking of these 50 features took 0.08 seconds on an 174 Mhz R10000 SGI O^2.

The memory requirements for feature tracking are two character images, and if desired for speed, two integer arrays for the gradients of the first image. The total memory for 1024x1024 images will be 10 MB with the gradient images or 2 MB without the gradient arrays.

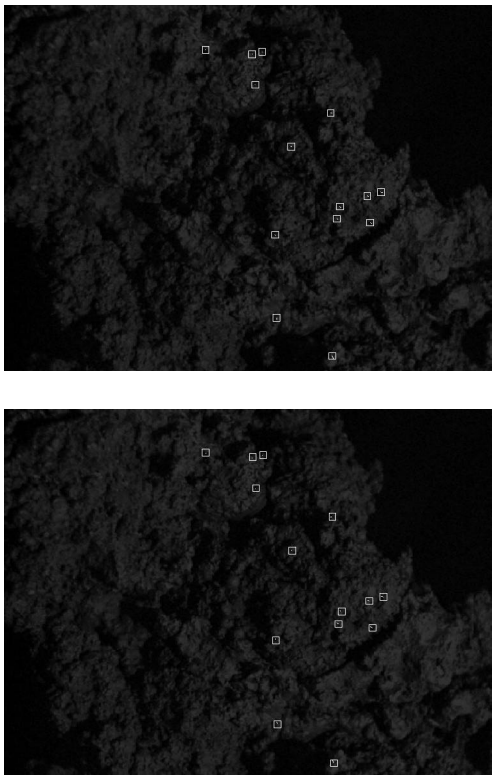


Figure 5: Tracked features. First image (top) and second image (bottom).

2.3 Two Frame Motion Estimation

The motion between two camera views can be described by a rigid transformation *(R,T)* where *R* encodes the rotation between views and $\boldsymbol{T}$ encodes the translation between views. Once features are tracked between images, the motion of the camera can be estimated by solving for the motion parameters that, when applied to the features in the first image, bring them close to the corresponding features in the second image.

In our algorithm, motion estimation is a two stage process. First an initial estimate of the motion is computed using a linear algorithm. This algorithm is applied multiple times using different sets of features to eliminate feature track outliers and determine a robust LMedS estimate of motion. The result of this algorithm is then used as input to a more accurate nonlinear algorithm that solves for the motion parameters directly. Since an good initial estimate is needed to initialize any nonlinear feature-based motion estimation algorithm, this two stage approach is common [22]. Output from the nonlinear algorithm is the estimate of the five motion parameters and their covariance. Our algorithm assumes that the camera taking the images has been intrinsically calibrated (i.e., focal length, radial distortion, optical center, skew and aspect are all known). Output from the nonlinear algorithm is the estimate of the five motion parameters and their covariance.

A fundamental short coming of all image-based motion estimation algorithms is the inability to solve for the magnitude of translational motion. Intuitively the reason for this is that the algorithms cannot differentiate between a very large object that is far from the camera or a small object that is close to the camera; the camera does not convey information about scene scale. Consequently, the output of motion estimation is a 5 DoF motion composed of the a unit vector $T_s = T/\|T\|$ describing the direction of heading and the rotation matrix *R* between views. As is shown in the next section, laser altimetry can be combined with 5 DoF motion estimation to compute the complete 6 DoF motion of the camera.

2.3.1. Robust Linear Motion Estimation

The first stage of motion estimation uses a linear algorithm to compute the motion between views[10]. Since the linear algorithm has a closed form solution, motion can be computed quickly. However, the linear algorithm does not solve for the motion parameters directly, so its results will not be as accurate as those obtained using the nonlinear algorithm. Our linear algorithm is an implementation of the algorithm presented in [21] augmented by normalization

presented in [6] for better numerical conditioning. To filter out possible outliers in feature detection, we use a robust linear motion estimation algorithm based on least median of squares[25]. Below we detail the computations in the algorithm; for a more complete description please see [6][21][25].

First, the homogenous coordinates of each feature are determined by projecting them onto the unit focal plane. This projection will depend on the lens, imager, and camera model used. A simple model for the transformation of a feature at pixel location (p_i,q_i) to its homogenous coordinates $\boldsymbol{u_i}$ is

$$\boldsymbol{u_i} = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} \dfrac{p_i - C_p}{f} \\ \dfrac{q_i - C_q}{sf} \\ 1 \end{bmatrix} \tag{9}$$

where (C_p,C_q) is the center of the camera in pixel units, f is the focal length of the camera in pixel units and s is the aspect ratio of the pixels. This model assumes no radial distortion in the camera. More sophisticated models that include radial distortion are used when necessary [20].

If $\boldsymbol{u_i}$ is the homogenous coordinates of feature i in the first image then let u'_i be the homogenous coordinates of the feature in the second image. The linear algorithm is based om the constraint that the optical centers of the two images and the 3-D location of the feature point must lie on a plane. This *epipolar constraint* can be written as

$$e_i = u'_i E u_i = 0 \tag{10}$$

where E is called the essential matrix and

$$E = [T_s]_x R \tag{11}$$

where $[\ ]_x$ signifies the cross product matrix

$$[(x_1, x_2, x_3)^T]_x = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}. \tag{12}$$

Equation 10 will hold for all features, but in the presence of feature position noise, it will not hold exactly. The linear algorithm utilizes these constraints and multiple features to solve for E by minimizing

$$min \sum_{i=1}^{N} u'_i E u_i \tag{13}$$

Using Equation 11, it then solves for the motion parameters.

To provide numerical stability [6], the homogenous coordinates from each image are first translated and scaled independently so that their centroid is (0,0,1) and their mean distance from the origin is $\sqrt{2}$. This can be accomplished by replacing each $\boldsymbol{u}_i$ with $\hat{\boldsymbol{u}}_i = Q\boldsymbol{u}_i$ where

$$Q = \begin{bmatrix} 1/k & 0 & -\bar{u}/k \\ 0 & 1/k & -\bar{v}/k \\ 0 & 0 & 1 \end{bmatrix} \qquad \bar{u} = \frac{1}{N}\sum_{i=1}^{N} u_i \qquad \bar{v} = \frac{1}{N}\sum_{i=1}^{N} v_i \tag{14}$$
$$k = \sum_{i=1}^{N} \sqrt{2((u_i - \bar{u})^2 + (v_i - \bar{v})^2)}$$

Each u'_i is replaced with $\hat{\boldsymbol{u}}'_i = Q'\boldsymbol{u}'_i$ using a similarly defined matrix Q'.

To solve Equation 13 the matrix

$$A = \begin{bmatrix} \hat{u}_1\hat{u}'_1 & \hat{u}_1\hat{v}'_1 & \hat{u}_1 & \hat{v}_1\hat{u}'_1 & \hat{v}_1\hat{v}'_1 & \hat{v}_1 & \hat{u}'_1 & \hat{v}'_1 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \hat{u}_N\hat{u}'_N & \hat{u}_N\hat{v}'_N & \hat{u}_N & \hat{v}_N\hat{u}'_N & \hat{v}_N\hat{v}'_N & \hat{v}_N & \hat{u}'_N & \hat{v}'_N & 1 \end{bmatrix} \tag{15}$$

is created. The solution $\boldsymbol{h}$ to $min\|A\boldsymbol{h}\|$ is the unit eigenvector of A^TA associated with the its smallest eigenvalue. This eigenvalue problem can be solved using standard algorithms like the **`jacobi`** algorithm from [14]. From $\boldsymbol{h}$, Q and Q', the essential matrix E is computed using

$$E = \begin{bmatrix} E_1 & E_2 & E_3 \end{bmatrix} = \sqrt{2}Q'^T \begin{bmatrix} h_1 & h_4 & h_7 \\ h_2 & h_5 & h_8 \\ h_3 & h_6 & h_9 \end{bmatrix} Q\,. \tag{16}$$

The robust linear motion algorithm uses the above equations and LMedS to find a solution to Equation 13 that is not influenced by outliers. The procedure, which is the same as the Least Median of Squares method for computing a robust estimate of the fundamental matrix in [25], is as follows. A subset of 8 feature tracks (the minimum number of points required) is selected at random from the set of all feature tracks. An estimate of the essential matrix is then computed using this subset and Equation 14 through Equation 16. Next, the epipolar error e_i from Equation 10 is computed for all feature tracks, and the median e_M of these errors is determined. If the median epipolar error for this subset is less than the median epipolar error determined from all of the previously selected subsets, the current essential matrix and its associated epipolar error become the best 8-point estimate for the essential matrix.

This process is repeated for a fixed number m of subsets that depends on the probability π of a sample free of outliers

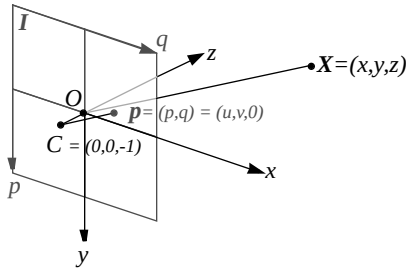


Figure 6: Unit focal length imaging geometry. World coordinate origin *O* is on image plane and optical center *C* is 1 unit behind image plane.

being selected and the percentage of outlier feature tracks ε in the set of feature tracks

$$m = \frac{\log(1-\pi)}{\log(1-(1-\varepsilon)^8)} \quad (17)$$

For our experiments, we set $\pi = 0.99$ and $\varepsilon = 20\%$ resulting in the selection of m=26 subsets (independent of the number of feature tracks).

Once the best 8-point estimate of the essential matrix is found, an additional step is required to estimate the robust essential matrix: the essential matrix computed from all of the feature tracks after outliers have been removed. First, the robust standard deviation is computed

$$\sigma_r = 1.4826\left(1 + \frac{5}{N-8}\right)\sqrt{e_M} \quad (18)$$

Next, outliers are detected by finding feature tracks whose square epipolar errors (using the best 8-point estimate of the essential matrix) are greater than $(2.5\sigma_r)^2$. Finally, the robust essential matrix is computed using all feature tracks that are not outliers according to the above criterion. This final estimate is the best robust estimate of the essential matrix because it takes into account all feature tracks while still eliminating feature track outliers.

The next stage in linear motion estimation is to extract the motion parameters from the essential matrix E. By manipulating Equation 13, it can be shown that $\boldsymbol{T_s}$ is the solution to $min\left\|E^T T_s\right\|$ which is the unit eigenvector with smallest eigenvalue of the matrix EE^T. Using the constraint that the scene must be in front of the camera (positive z), the sign of the translation can be determined. If

$$\sum_i (T_s \times \boldsymbol{u}'_i) \cdot (E\boldsymbol{u}_i) < 0 \quad (19)$$

then $T_s \Leftarrow -T_s$ where the summation is over a few randomly selected 3-D points to ensure robustness to noise.

Finding the solution to the rotation matrix R is more involved. Rearranging Equation 11 results in $R^T[-T_s]_x = E^T$; given that there exists noise in the feature tracks, R can be found by solving

$$min\left\|(R^T[-T_s]_x - E^T)\right\| \quad (20)$$

subject to R being a rotation matrix. Equation 20 can be solved for R^T as follows: Let

$$\begin{aligned} C &= \begin{bmatrix} C_1 & C_2 & C_3 \end{bmatrix} = [-T_s]_x \\ D &= \begin{bmatrix} D_1 & D_2 & D_3 \end{bmatrix} = E^T \\ B &= \sum_{i=1}^{3} B_i^T B_i \qquad B_i = \begin{bmatrix} 0 & (C_i - D_i)^T \\ D_i - C_i & [D_i + C_i]_x \end{bmatrix} \end{aligned} \quad (21)$$

then the eigenvector corresponding to the smallest eigenvalue of B is the unit quaternion q associated with R^T. Equation 22 can then be used to transform between the quaternion and matrix representations of R.

2.3.2. Nonlinear Motion Estimation

Robust linear motion estimation serves two purposes: it provides an initial estimate of the 5 DoF motion between views and it detects and eliminates feature track outliers. The nonlinear algorithm takes the initial linear estimate of the motion and refines it by minimizing an error term that is a function of the motion parameters and the outlier-free feature tracks. There exists many nonlinear motion estimation algorithms in the vision literature. Instead of starting from scratch, the nonlinear algorithm we have developed combines the attractive elements of multiple algorithms to produce an algorithm that is computationally efficient, numerically stable and accurate. For numerical stability, we use the camera model parameterization of Azarbayejani and Pentland[1]. For highly accurate motion parameter estimation we use the Levenberg-Marquardt algorithm as proposed by Szeliski and Kang[18]. Finally, for computational efficiency, we remove the scene structure from the nonlinear minimization as suggested by Weng et al. in [22].

Before we can express the error function, we need to detail the motion parameters over which the minimization will take place. First of all, the motion between frames is presented as a translation and rotation pair *(R,T)*. To simplify the parameter estimation, we represent the rotation with a unit quaternion $q = \begin{bmatrix} q_o & q_1 & q_2 & q_3 \end{bmatrix}^T$ where the rotation matrix in terms of a unit quaternion is

$$R(q) = \begin{bmatrix} q_o^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_o^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_o^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \quad (22)$$

The translation is represented by a unit vector $T = \begin{bmatrix} T_x & T_y & T_z \end{bmatrix}^T$. Together the unit quaternion and unit translation comprise the parameter state vector $\boldsymbol{a}$.

$$\boldsymbol{a} = \begin{bmatrix} q_o & q_1 & q_2 & q_3 & T_x & T_y & T_z \end{bmatrix}^T \quad (23)$$

Nonlinear motion estimation attempts to minimize the image plane error between the features in the second view and the projection of the features in the first view into the second view given the motion between frames. In the photogrammetry, this technique is called bundle adjustment. If the unit focal coordinates (defined by Equation 9) of the features in image I are $\boldsymbol{u}_i = \begin{bmatrix} u_i & v_i \end{bmatrix}^T$ and $\boldsymbol{u}_i' = \begin{bmatrix} u_i' & v_i' \end{bmatrix}^T$ in image J, then the image plane error is

$$C(a) = \sum_i \left\| u_i' - f(\boldsymbol{u_i}, \boldsymbol{a}) \right\|^2 \quad (24)$$

where f represents the projection of the features $\boldsymbol{u}_i'$ into image J given the motion $\boldsymbol{a}$. Correct image projection requires knowledge of the depth to a feature and a perspective camera model. Using the model of Azarbayejani and Pentland [1], if the (unknown) feature

depths from the image plane are z_i, then the relation between unit focal feature coordinates and 3-D feature coordinates is

$$X_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} u_i(1+z_i) \\ v_i(1+z_i) \\ z_i \end{bmatrix} \quad (25)$$

The features in image I are transformed into image J according to

$$X'_i = \begin{bmatrix} x'_i & y'_i & z'_i \end{bmatrix}^T = R(\boldsymbol{q})X_i + T. \quad (26)$$

By combining Equation 25 and Equation 26, the feature depths $\begin{bmatrix} z_i & z_i' \end{bmatrix}^T$ can be computed through triangulation by solving

$$\begin{bmatrix} -R\boldsymbol{u_i} & \boldsymbol{u_i'} \end{bmatrix} \begin{bmatrix} z_i \\ z_i' \end{bmatrix} = T \quad (27)$$

assuming that the translation between views is nonzero[21].

The camera model given the imaging geometry, shown in Figure 6, is

$$f(\boldsymbol{u'_i}, \boldsymbol{a}) = \begin{bmatrix} x' \\ y' \end{bmatrix} \frac{1}{1+z'} \quad (28)$$

Combining Equation 25 Equation 26 and Equation 28 results in a complete definition of Equation 24.

To estimate the motion parameters, we minimize Equation 24 using the Levenberg-Marquardt algorithm for nonlinear minimization. This approach was also used by Szeliski and Kang [18], however, unlike in their approach, we do not include the feature depths in the minimization. The result is an accurate and computationally efficient approach to motion estimation.

The Levenberg-Marquardt algorithm finds the minimum of $C(\boldsymbol{a})$ by iteratively solving

$$(A + \lambda I)\delta\boldsymbol{a} = -\boldsymbol{b} \quad (29)$$

for $\delta\boldsymbol{a}$ where

$$A = \sum_i \left(\frac{\partial}{\partial a} f(\boldsymbol{u_i}, \boldsymbol{a})\right)^T \left(\frac{\partial}{\partial a} f(\boldsymbol{u_i}, \boldsymbol{a})\right), \quad (30)$$

$$\boldsymbol{b} = \sum_i \left(\frac{\partial}{\partial a} f(\boldsymbol{u_i}, \boldsymbol{a})\right)(u_i' - f(\boldsymbol{u_i}, \boldsymbol{a})) \quad (31)$$

and λ is a scalar whose value is changed at each iteration depending on the solution to Equation 29. After each iteration, the estimate of $\boldsymbol{a}$ is updated to $\boldsymbol{a}+\delta\boldsymbol{a}$.

In our application, $\frac{\partial f}{\partial a}$ is a 2x7 matrix

$$\frac{\partial}{\partial a} f(\boldsymbol{u_i}, \boldsymbol{a}) = \left[\frac{\partial}{\partial q} f(\boldsymbol{u_i}, \boldsymbol{a}) \; \frac{\partial}{\partial T} f(\boldsymbol{u_i}, \boldsymbol{a})\right] \quad (32)$$

where, using chain rule,

$$\frac{\partial}{\partial q} f(\boldsymbol{u_i}, \boldsymbol{a}) = \frac{\partial}{\partial X'} f(\boldsymbol{u_i}, \boldsymbol{a}) \frac{\partial}{\partial q} X' \quad (33)$$

and

$$\frac{\partial}{\partial T} f(\boldsymbol{u_i}, \boldsymbol{a}) = \frac{\partial}{\partial X'} f(\boldsymbol{u_i}, \boldsymbol{a}) \frac{\partial}{\partial T} X'. \quad (34)$$

From Equation 28

$$\frac{\partial}{\partial X'} f(\boldsymbol{u_i}, \boldsymbol{a}) = \begin{bmatrix} \frac{1}{1+z'} & 0 & -\frac{x'}{(1+z')^2} \\ 0 & \frac{1}{1+z'} & -\frac{y'}{(1+z')^2} \end{bmatrix} \quad (35)$$

The technique presented in Wheeler and Ikeuchi [23] can be used to simplify jacobian of rotation by minimizing about the identity quaternion $\boldsymbol{q_I}$. Equation 26 can be rewritten as

$$X' = R(\boldsymbol{q}_I)R(\boldsymbol{q})X + \boldsymbol{T} = R(\boldsymbol{q}_I)\tilde{X} + T \quad (36)$$

using the fact that $R(\boldsymbol{q_I})$ is the identity and $\tilde{X} = R(\boldsymbol{q})X$. Following the derivation in [23],

$$\frac{\partial}{\partial q} X' = \frac{\partial}{\partial q} R|_{q=q_I} R(\boldsymbol{q})X = \frac{\partial}{\partial q} R|_{q=q_I} \tilde{X} = \begin{bmatrix} 0 & 0 & 2\tilde{z} & -2\tilde{y} \\ 0 & -2\tilde{z} & 0 & 2\tilde{x} \\ 0 & 2\tilde{y} & -2\tilde{x} & 0 \end{bmatrix} \quad (37)$$

The advantage of this form of the jacobian of rotation is that is enforces the unit magnitude constraint on the quaternion and its simple form results in efficient computation.

The jacobian of translation is

$$\frac{\partial}{\partial T} X' = I \quad (38)$$

Using Equation 32 through Equation 38, $\frac{\partial}{\partial a} f(\boldsymbol{u_i}, \boldsymbol{a})$ can be determined, so A and $\boldsymbol{b}$ can be computed.

The Levenberg-Marquardt algorithm (a brief description is given in Numerical Recipes in C [14]) minimizes nonlinear functions by continuously varying between steepest descent far from the minimum and the inverse-Hessian method close to the minimum. Variation between minimization methods is controlled by the value of λ in Equation 29; small values of λ correspond to inverse-Hessian and large values correspond to steepest descent.

1. *Compute* $C(\boldsymbol{a})$
2. *Set* $\lambda = 0.001$ *(start with inverse-Hessian)*
3. *Compute* A,$\boldsymbol{b}$ *and solve* $(A + \lambda I)\delta\boldsymbol{a} = -\boldsymbol{b}$ *for* δa.
4. *Evaluate* $C(\boldsymbol{a} + \delta\boldsymbol{a})$ *with* $\|\boldsymbol{q} + \delta\boldsymbol{q}\| = 1$ & $\|\boldsymbol{T} + \delta\boldsymbol{T}\| = 1$
5. *If* $C(\boldsymbol{a} + \delta\boldsymbol{a}) \geq C(\boldsymbol{a})$

   $\lambda \leftarrow 10\lambda$

   goto 3.

   Else

   $\lambda \leftarrow 0.1\lambda$

   $\boldsymbol{q} \leftarrow (\boldsymbol{q} + \delta\boldsymbol{q})/\|\boldsymbol{q} + \delta\boldsymbol{q}\|$

   $\boldsymbol{T} \leftarrow (\boldsymbol{T} + \delta\boldsymbol{T})/\|\boldsymbol{T} + \delta\boldsymbol{T}\|$

   $\boldsymbol{a} \leftarrow \boldsymbol{a} + \delta\boldsymbol{a}$

   solve for new feature depths using Equation 27

   goto 3.

Stop when $(C(\boldsymbol{a} + \delta\boldsymbol{a}) - C(\boldsymbol{a}))/C(\boldsymbol{a}) < 0.001$

Figure 7: Nonlinear motion estimation algorithm.

Since we are solving for a rotation represented by a unit quaternion and also a unit length translation, these constraints need to be enforced during minimization. We enforce these constraints by setting $\|\boldsymbol{q} + \delta\boldsymbol{q}\| = 1$ and $\|\boldsymbol{T} + \delta\boldsymbol{T}\| = 1$ during the update of the parameter vector at each iteration. Consequently, these constraints are enforced while not complicating the minimization by including the constraints explicitly in the minimization function.

The complete nonlinear minimization procedure is given in Figure 7.

In our implementation of nonlinear motion estimation, the scene structure, encoded in the relative depths, is not included in the parameter vector during minimization. Inclusion of the feature depths would increase the length of the parameter vector from 7 to 7+N. Since the minimization relies on an inversion of a square matrix of rank equal to the length of the parameter vector to solve Equation 29, a computationally expensive matrix inversion would result. Since feature depths can be computed directly from the motion between views, it is not necessary to include them in the parameter vector. Instead, at each iteration, the feature depths are updated using the current motion estimate. The result is a computationally efficient and accurate motion estimation algorithm. It should be noted that in the case of multi-frame motion estimation, the inclusion of structure in the state is recommended because it enforces consistency in motion and structure estimates across multiple frames containing the same set of features. Since we are computing motion for just two frames, it is not necessary for our application.

The output of nonlinear motion estimation is an estimate of the 5 DoF motion between views. In addition, the covariance Σ of the motion parameters $\boldsymbol{a}$ can be extracted directly from the quantities computed during minimization using

$$\Sigma(\boldsymbol{a}) = A^{-1}. \tag{39}$$

2.3.3. Processing Issues

If the number of features *N* is much greater than 8, then the running time of closed form linear motion estimation is *O(N)*. However, for the robust linear motion estimation algorithm, the number of features used in each trial is fixed at 8, so each trial takes constant time. Therefore, the running time of robust linear motion estimation is *O(m)* proportional to the number of trials m and does not depend on the number of features tracked. For the images shown in Figure 5, the robust linear motion algorithm took 0.63 seconds for 78 trials (π =0.99, ε = 0.3) on an 174 Mhz R10000 SGI O^2.

Each iteration of the nonlinear motion algorithm is *O(N)*, so given *I* iterations, the nonlinear algorithm is *O(NI)*. For the two images shown in Figure 5, the nonlinear motion algorithm took 0.18 seconds for 50 features and 4 iterations on an 174 Mhz R10000 SGI O^2.

For both motion algorithms, the memory requirements for motion estimation are linear in the number of features tracked. Since the number of features is much smaller than the size of the images in pixels, the memory required for motion estimation will be much less than that required for feature detection and tracking.

2.4 Scale Computation Using Altimeter

The final stage of motion estimation computes the remaining motion parameter, magnitude of translation, from laser altimetry data. Depending on descent angle and surface relief, one of two complimentary methods is used.

2.4.1. Motivation

Motion estimation using monocular imagery cannot solve directly for the magnitude of translation, so an external means must be used to recover this parameter. For a spacecraft in orbit about a small body, there exist multiple possible solutions.

One solution is to integrate the accelerometer measurement in the spacecraft inertial reference unit to determine position. The advantage of accelerometers is that they present a completely onboard solution. Unfortunately, because that come from integration of noisy acceleration measurements, position measurements from accelerometers are too inaccurate for precision landing. For example, if the accelerometer measurement has an error of 10 μg, the position error can grow to the kilometer level in an hour.

The traditional approach is to use radiometric tracking measurements from earth. This approach has the advantage that it is well understood and uses equipment already on board the spacecraft. However, radiometric tracking has many disadvantages. First, it requires dedicated Deep Space Network tracking which is expensive and at times difficult to schedule. Round trip light time for tracking from earth induces a large latency in any position measurements (approximately 24 minutes for comet Tempel 1). At deep space distances, radiometric tracking is not accurate enough relative to the size of maneuvers needed to explore a small body.

Another option is to perform radiometric tracking between the orbiter and the lander during precision landing [19]. This is a well understood and accurate technology. However, it has some distinct disadvantages: it requires a line of sight between the orbiter and lander; the orbiter position must be known accurately; the lander must be tracked from the orbiter; and additional equipment is needed on the lander and orbiter.

In a similar vein, surface beacons in known position deposited by the spacecraft before landing can be used for position estimation. Positioning based on beacons can be very accurate, however, if 3-D position is desired, then

multiple beacons must be deployed and all beacons must be in line of sight with the lander at all times during descent. The major disadvantage of this approach is that the beacons must deployed and anchored to the surface and the possibly massive beacon deployment mechanism must be added to the spacecraft payload.

As shown in Table 1, multiple missions have or are using laser altimeters for science return and navigation. Table 1 also shows that as the design of laser altimeters progresses, their size, weight and power consumption are decreasing while there accuracy and speed are increasing. As shown below, laser altimeters can also be used as a navigation sensor by aiding the determination of the position of the spacecraft. Laser altimeters give accurate range estimates and, when combined with a descent imager, present a complete on-board solution to 6-D body relative motion estimation. A disadvantage of the laser altimeter approach is that they have limited range (50 km for the NEAR laser altimeter). However, near body operations is precisely when accurate position estimation is needed the most, so this is not a major issue. A laser altimeter is an additional sensor, however, science return combined with navigational use justify the addition. Based on the disadvantages of the other available options, we determined that the use of a laser altimeter was the most promising solution for scale estimation.

Another promising but immature technology that could be used for position estimation is automatic recognition of small body landmarks for determination of absolute position. The advantages of this approach are that it requires no additional equipment (a camera is sufficient) and it is a stand-alone, on-board solution. Currently, robust and completely autonomous solutions to this problem have not been developed, however many approaches appear promising [9]. For this option to become a viable solution, the landmark recognition and position estimation algorithms will have to be efficient, accurate and robust.

2.4.2. Difference Scale Estimation

If images are taken as the spacecraft descends vertically to the surface, or the surface has very little surface relief, computation of translation magnitude is straightforward. Laser altimeter readings A_I and A_J are acquired simultaneously with each image. As shown in Figure 8, the difference in altimeter readings is equal to the translation of the spacecraft along the z-axis between images. Consequently, the magnitude of translation is

$$\|T\| = \frac{(A_I - A_J)}{t_z} \tag{40}$$

For motion approaching horizontal, t_z approaches zero, Equation 40 becomes ill conditioned and difference scale estimation will not work. Furthermore, if the spacecraft is not descending vertically and the surface topography is rough on order of the scale of translation then the difference of altimeter readings will not accurately reflect the z component of the translation. Once again, difference scale estimation will not work. Fortunately a different, albeit more complicated, procedure exists for computing scale in these cases.

2.4.3. Structure-Based Scale Estimation

From the feature-based motion estimate, the scaled depths α_i (Equation 27) to features in the scene can be computed. Assuming, without loss of generality, that the laser altimeter is aligned with the camera optical axis, features in the center of the image will be at a depth equivalent to the laser altimeter reading. Consequently, the ratio of the laser altimeter reading to the scaled feature range will be the magnitude of translation. This approach requires only one altimeter reading, so it is not susceptible to errors from changing surface relief. Furthermore, it does not depend on nonzero translation along the z-axis. In fact, structure-based scale estimation works better when the

Table 1: Laser Altimeters

mission	name	use date	sample rate	max range	accuracy	size	mass	power
Clementine	Clementine LIDAR[13]	1994	1 Hz	640 km	40 m	13x15x4 cm[a] + 17x18x36 cm	1.2 kg[a] + 1.1 Kg	6.8 W[a] + 9.5 W
Mars Global Surveyor	Mars Orbiter Laser Altimeter [17]	1999	10 Hz	786 km	2.0 m	50x50x75cm	25 kg	34W
Near Earth Asteroid Rendezvous	NEAR Laser Altimeter [5]	2000	8 Hz	50 km	2.0 m	37x23x22cm	5kg	15W
DS-4/ Champollion	Laser Radar Instrument[b]	2006	10,000 Hz	2 km	0.2m	20x10x10cm	2 kg	10W

a. The Clementine LIDAR has two parts: the laser transmitter (first spec.) and the HIRES camera for receiver optics (second spec.).
b. The LRI figures are design specifications based on a JPL internal design document for the scanning laser rangefinder being built for DS-4/Champollion.

spacecraft is descending at an angle with respect to the surface because in this case, scene structure can be estimated more accurately than for pure descent.

The procedure for structure-based scale estimation is to first compute the feature based motion between images along with the depth of the features in the image (a by-product of nonlinear motion estimation). Assuming alignment of laser altimeter with the optical axis, the features near the center of the image will be geometrically close to the surface patch that supplies the reading for the laser altimeter (see Figure 8). Since it is unlikely that a feature will correspond exactly to the image center, a few (3-5) features closest to the image center are selected and weighted interpolation is used to determine the scene depth at the image center α_c

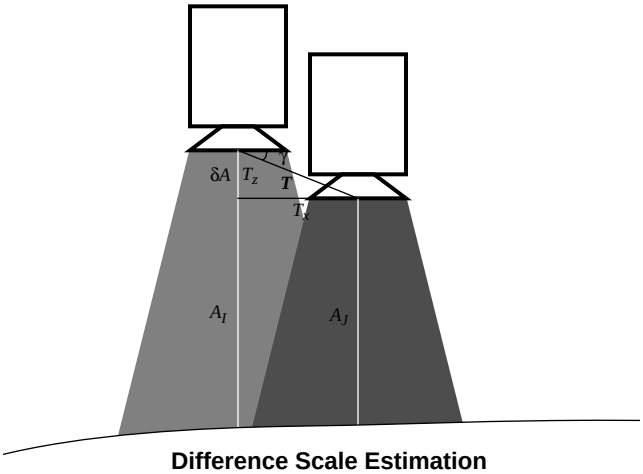
$$\alpha_c = \left(\sum_j \frac{\alpha_j}{\|(u_j, v_j)\|}\right) \Big/ \left(\sum_j \frac{1}{\|(u_j, v_j)\|}\right) \quad (41)$$

In the above equation, the feature depths are weighted by $\|(u_j, v_j)\|$ which is proportional to the distance from the image center; features closer to the image center will have greater weight. The image-based scene depth at the image center has the same depth as the altimeter reading taken when the first image was acquired, so the magnitude of translation is

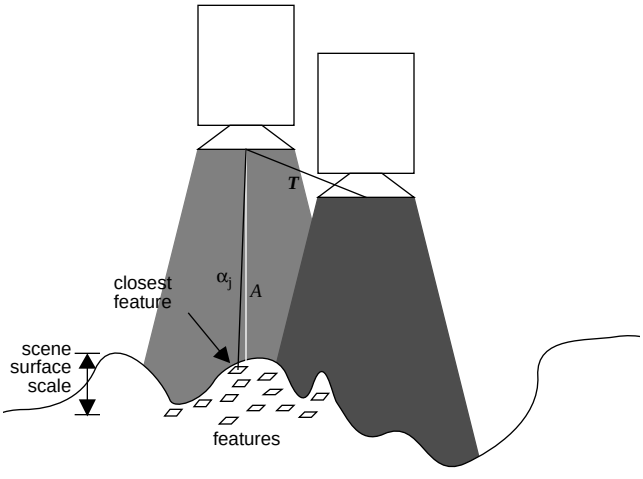
$$\|T\| = \frac{A_I}{\alpha_c} \quad (42)$$

A number of observations can be made about structure based scale estimation. First, As the translation between images approaches vertical, the structure estimates degrade, especially near the optical axis (i.e., on the optical axis, the displacement between features will be zero for vertical descent - structure from triangulation cannot be computed). Fortunately, vertical descent is precisely the motion where difference scale estimation works best. The complementary nature of these two scale estimation algorithms will be explored further in Section 4. Second, for the altimeter reading to be related to scene structure, a feature must be located near the optical axis in the first frame. Consequently, structure-based scale estimation will work better when more features are tracked in order to guarantee that a feature will be near the optical axis. Alternatively, another approach is to always make the image center a feature so that a depth value will always exist near the altimeter reading. Since this approach requires that the image center have sufficient structure for feature tracking, which cannot be guaranteed, it was not attempted.

The magnitude of translation from laser altimetry when combined with feature-based motion completes the 6 DoF motion estimation of the spacecraft.



Difference Scale Estimation



Structure Scale Estimation

Figure 8: Methods for estimating translation magnitude.

2.5 Multi-frame Issues

The algorithm presented above solves for 6 DoF motion using feature tracking between two image frames. Given an image stream (multiple images taken in a sequence), it is possible to track features across multiple images; this leads to faster and more robust feature tracking and ultimately more accurate motion estimation. Consequently, when estimating motion for an image stream, we track features across multiple images which requires some minor modifications to our algorithm.

The primary advantage of multi-frame feature tracking is that features do not have to be detected for each frame. By definition, the features that were detected in the first frame will remain features, image locations that are easy to track, for many images after they were detected. Consequently, a feature detected in the first frame is tracked until the appearance of the feature changes too much for continued tracking (occlusion, passes out of field of view, large viewing or illumination change). Eventually, too many features will be eliminated for precise motion estimation, so new features will have to be detected. In our

implementation, features are added at key frames in the sequence. The number of features added is such that the total of the features still being tracked at the key frame and those added equals the number of features tracked in the first frame. For the results presented in this paper, key frames occur every 4 images in the stream. Since features do not have to be detected every image, but every key frame, the average processing time per frame is reduced. Note that with multi-frame tracking, the features for the current frame could have been added at any previous key frame, so our feature tracking data structures are designed to handle arbitrary frame starting points for features.

Feature tracking across multiple frames is the same as feature tracking across two frames with a few modifications. First, the feature window (image *I*) is taken from the image where the feature was first detected and the tracking window is taken from the current image (image *J*). This will prevent feature drift by guaranteeing that the feature window being tracked is the one originally detected. Second, the feature tracking displacement $\boldsymbol{d}$ is initialized at the feature tracking displacement computed for the previous frame instead of being initialized to $\boldsymbol{0}$. This will reduce the number of iterations in for tracking each feature making feature tracking more efficient.

If it can be assumed that the motion of the spacecraft is smooth (C1), then the trajectory of the features through the image stream will be locally linear (i.e., the feature positions will lie along a line). This assumption leads to two possible improvements to feature tracking. To define locally linear, suppose that the image position of a feature in the three most recent frames are $\boldsymbol{p}_{i-2} = (u_{i-2}, v_{i-2})$, $\boldsymbol{p}_{i-1} = (u_{i-1}, v_{i-1})$ and $\boldsymbol{p}_i = (u_i, v_i)$. A feature track is locally linear if

$$\operatorname{acos}\frac{(\boldsymbol{p}_{i-2} - \boldsymbol{p}_{i-1}) \cdot (\boldsymbol{p}_{i-1} - \boldsymbol{p}_i)}{\|\boldsymbol{p}_{i-2} - \boldsymbol{p}_{i-1}\| \|\boldsymbol{p}_{i-1} - \boldsymbol{p}_i\|} < \Lambda . \tag{43}$$

where Λ is an angular threshold that we set to $\pi/6$. If a feature track is not locally linear than it is likely that the tracking is erroneous, so the feature is eliminated. This feature filtering reduces the number of feature track outliers *before* motion estimation making motion estimation more robust. It also reduces the percentage of outlier feature tracks ε, so the number of feature track subsets that have to be investigated during robust linear motion estimation (Equation 17) is reduced leading to a reduction in the time spent in the linear motion algorithm.

Given the smooth trajectory assumption, the position of the features in the next frame can be predicted. Extrapolating the notation from above, the predicted position of the feature $\boldsymbol{p}_{i+1}$ in the next frame can be expected to be close to the linear extrapolation of the feature positions in the previous two frames.

$$\boldsymbol{p}_{i+1} = \boldsymbol{p}_i + (\boldsymbol{p}_i - \boldsymbol{p}_{i-1}) \tag{44}$$

If the extrapolated position is used to initialize feature tracking, then the feature track position will start out very close to its true position. This will result in less iterations and time spent during feature tracking.

Furthermore, given the smooth trajectory assumption, it is possible to eliminate the robust linear motion estimation algorithm altogether. Local linearity in feature tracks can be used to eliminate outliers feature tracking errors. Instead of the linear algorithm estimate, the motion estimated from the previous frame can be used as the initial estimate for the nonlinear motion estimation algorithm. With a way to eliminate outliers and provide an initial estimate to the nonlinear algorithm the linear algorithm is no longer needed. However, to be safe, we still include the linear motion estimation algorithm in our multi-frame implementation.

Another advantage of tracking features across multiple frames is that the disparity between the feature detection and current image positions increases. As disparity increases, the sensitivity of motion estimation to feature tracking errors decreases, so the errors in motion estimation will decrease. We show in Figure 12 that when modified to compute motion between the initial and current frame, our multi-frame algorithm is able to compute more accurate motion estimates. To properly handle multi-frame motion estimation, we need an algorithm that handles features added at different key frames. We are currently investigating this issue.

2.6 Discussion

The algorithm we have presented can be used to estimate motion with respect to any proximal surface. Consequently, it can be used for precision landing on comet nuclei. asteroids and small moons. It can also be used for proximity operations during rendezvous and docking between two spacecraft. Another application is estimating the attitudinal motion of a orbiter or satellite during precision pointing to surface targets. Rotational motion is completely determined from image-based motion estimation, so a laser altimeter is unnecessary for this application.

One of the underlying assumptions of the algorithm is that the surface being imaged has texture. Texture is needed for feature detection and tracking. Comet surfaces are expected to be rough at all scales, so shading variation due to changing surface normal will provide adequate texture for feature tracking. Asteroid surfaces will have some regions of little texture, but because of cratering, there should be sufficient texture in parts of the image for feature tracking (see Figure 2). In other applications, the texturedness of the surfaces to be imaged should be guaranteed before this algorithm is selected for motion estimation.

Our motion estimation works best when the spacecraft is close to the surface being imaged. In general image based motion estimation algorithms are more accurate when

differences in body shape or terrain height are observable between image frames. This generally occurs when the spacecraft is close to the surface. Another advantage of being close to the surface is that the surface fills the field of view, so features can be tracked in the entire image. For a given camera field of view, the wider the spread in feature tracks, the more accurate the motion estimation becomes.

In feature-based motion estimation, there are couplings between translational and rotational motions. For example, a rotation about the *X* axis will cause the features to move in the image in much the same way as a translational motion along the *Y* axis. This coupling cannot be avoid, however, it can be detected. The covariance of the image-based motion estimate (Equation 39) will encode this coupling, so that when the motion estimate and covariance are incorporated into the spacecraft guidance, navigation and control filter, these dependencies will not bias the estimate of spacecraft motion from all onboard sensors.

3 Results on Real Imagery

To test our motion estimation algorithm, we generated two sequences of real imagery. First a comet nucleus analog was created by a comet scientist at JPL. This analog is rough at all scales and matte black, the expected characteristics of comet nuclei. The analog has an approximate diameter of 25 cm. We placed the analog on a rigid stand and took two sequences of images as the camera moved toward the comet analog. The first sequence which we call *descent* was with a 640x480 CCD imager, a 15 degree field of view lens. The second sequence called *approach* was taken with a 1024x1024 CCD imager and a 25 degree field of view lens. Both sequences were acquired with the camera starting 80 cm from the comet analog; the camera moved 1.00 cm toward the analog between each image.

Ground truth for the image sequence motions were obtained though camera calibration [20]. Each camera was calibrated using a calibration target and as a by product of the calibration procedure, the direction of translation was computed. For the descent sequence, the true translation direction is (0,0,-1), and for the approach sequence, the true translation direction is (0.0096, -0.0033, -0.9999). Since the cameras were rigidly fixed, there was no rotation in the motion.

An altimeter reading was simulated for each image by using the translation stage reading as the altimeter reading. Using this data type, the scale of translation is know to the accuracy of the translation stage, so no scale estimation method is needed. In the future, an imaging setup with a laser altimeter will be used to test the scale estimation methods.

The results are presented in the Figure 9 through Figure 13. For each sequence, motion is estimated using 50 or 500 features. At the top of each figure is shown the feature tracks for the entire sequence. Different color tracks correspond to the different key frames when the features were added to the sequence; a key frame occurred every 4 frames. Next are shown the computed translation *(tx,ty,tz)* and rotation angles *(rx,ry,rz)* of the motion computed for each frame using the two stage motion estimation algorithm. Following these is a plot showing the translation error magnitude (vector distance between the true and estimated translations) for each frame in the sequence. On this plot, the dashed line corresponds to the expected performance of the algorithm established using Monte Carlo simulation (assuming perfect feature tracking) for the imaging parameters and motion (See Section 4). Finally, the rotation error magnitude (vector difference between estimated and true rotation angles) is shown for each frame. Again, the dashed line corresponds to the expected performance of the algorithm established using Monte Carlo simulation. Table 2 summarizes the results from the

Table 2: Motion estimation results.

sequence	number of features	motion estimation stages	δT_{seq} (cm)	δR_{seq} (degrees)	processing time (seconds)	number of frames	frame rate (Hz)	δT_{sim} cm)	δR_{sim} (degrees)
descent	50	linear	0.044927	0.06376	6.24	25	4.01		
descent	50	nonlinear	0.044966	0.0662209	13.1	25	1.90	0.0579763	0.0411912
descent	500	linear	0.033483	0.056666	31.61	25	0.79		
descent	500	nonlinear	0.033615	0.056834	82.33	25	0.30	0.0169	0.0120
approach	50	linear	0.028092	0.024439	2.4	7	2.91		
approach	50	nonlinear	0.023936	0.021443	3.94	7	1.77	0.0659696	0.0505746
approach	500	linear	0.01861	0.017992	13.42	7	0.52		
approach	500	nonlinear	0.018938	0.15937	24.05	7	0.29	0.0221996	0.169442

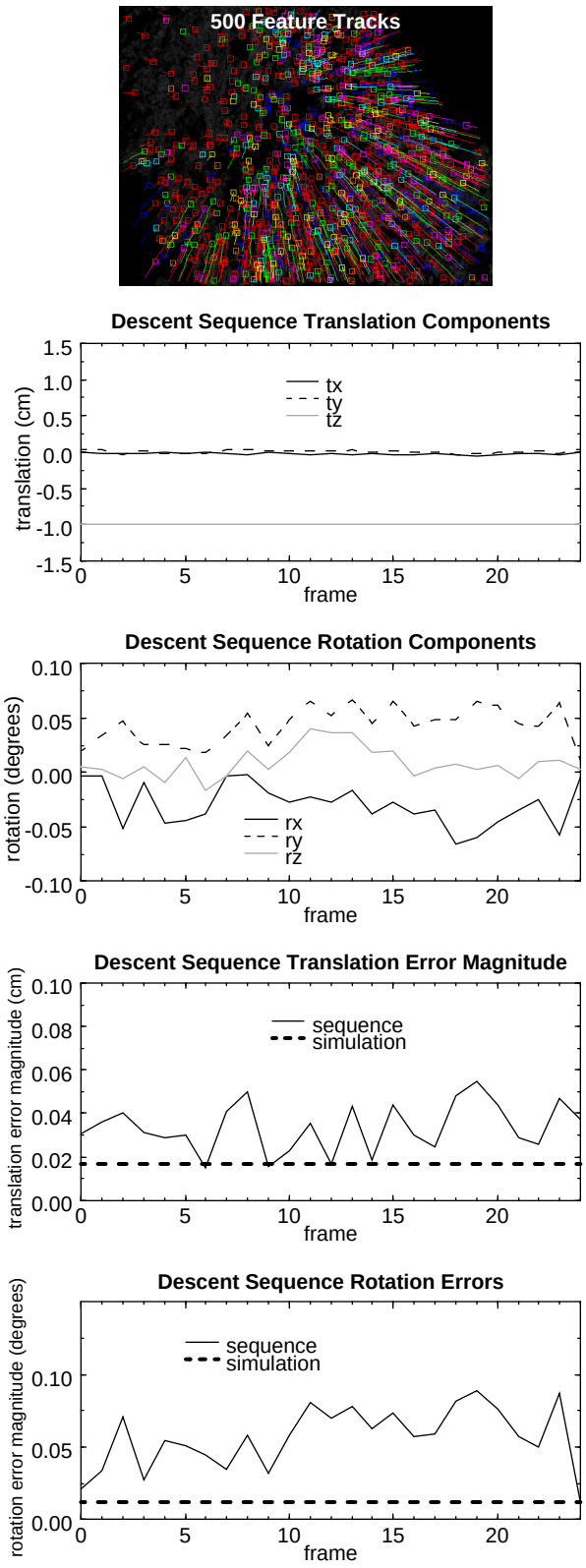


Figure 9: Motion Estimation for the Descent Sequence with 500 features tracked.

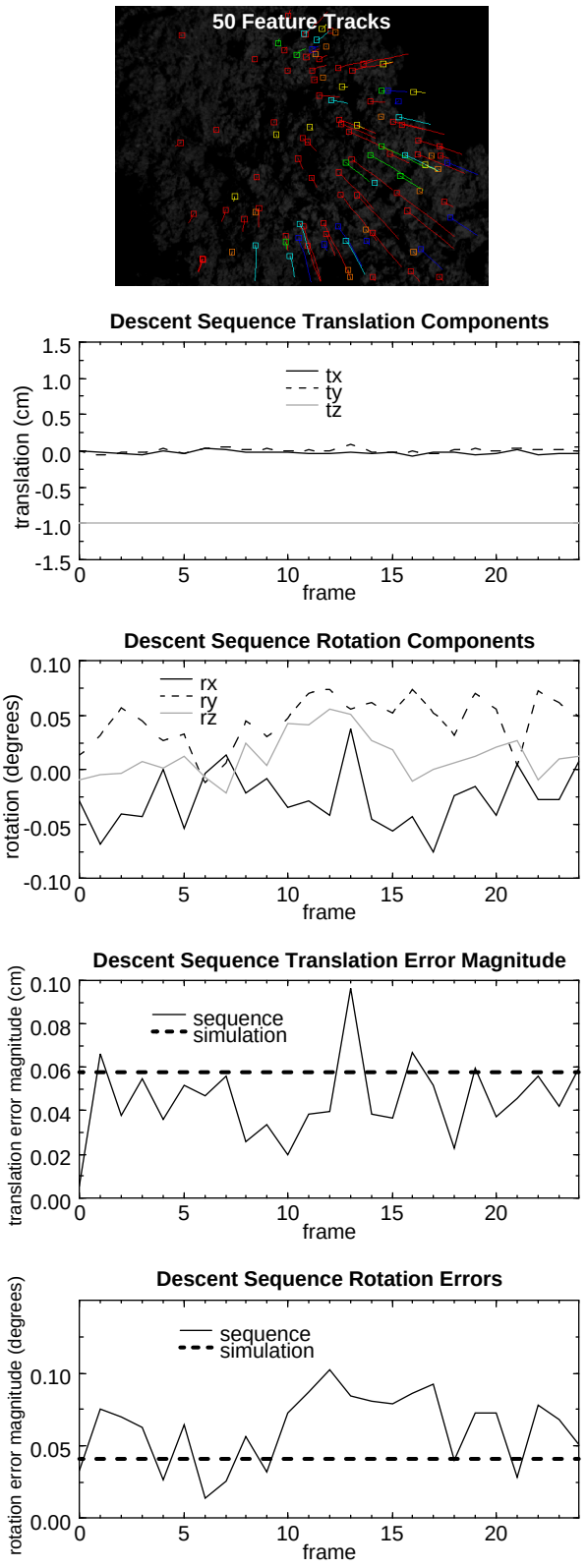


Figure 10: Motion Estimation for the Descent Sequence with 50 features tracked.

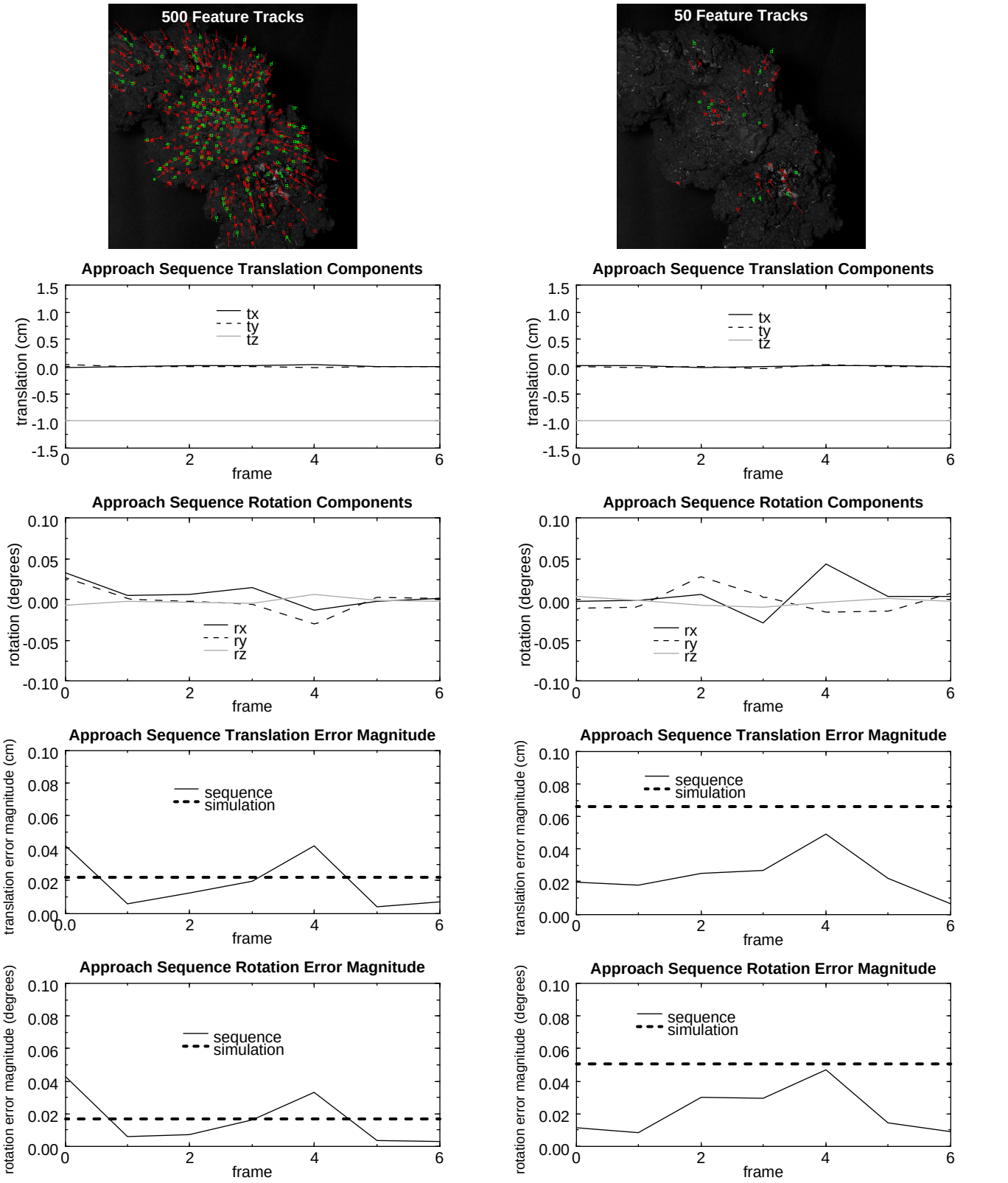


Figure 11: Motion Estimation for the Approach Sequence with 500 features tracked.

Figure 12: Motion Estimation for the Approach Sequence with 50 features tracked.

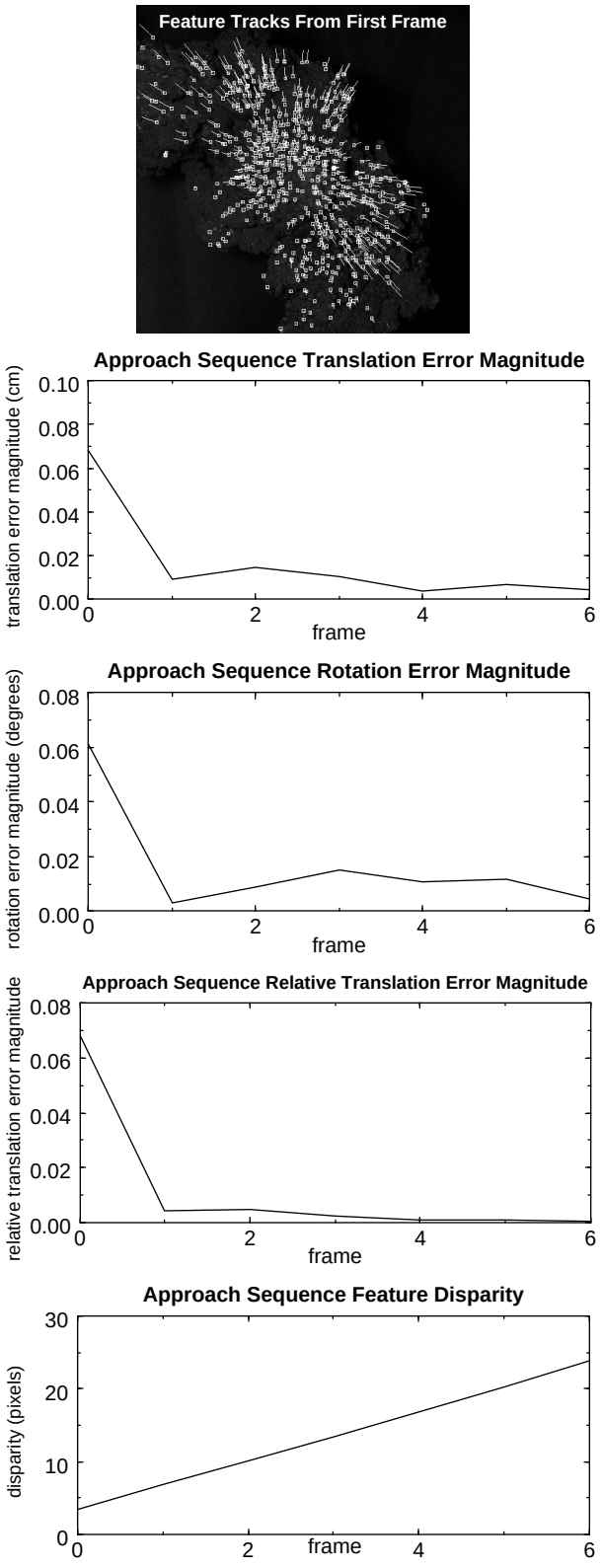


Figure 13: Motion estimated from the first frame for the Approach Sequence. Shows as feature disparity increases, the motion estimates improve.

four sequences and compares the motion estimation results for the linear algorithm to the motion estimation results obtained using the linear and nonlinear algorithm.

The results in Table 2, show that in general the addition of the nonlinear motion estimation algorithm does not improve the results of motion estimation all that much. This is because for vertical descent, the motion computed using the linear algorithm is very constrained, so the results are very close to those obtained using the nonlinear algorithm. Including the nonlinear algorithm in general doubles the running time of the algorithm, so for the vertical descent, it is probably a good idea to remove this stage from the algorithm if running time is important.

Another general trend shown in Table 2 is that changing from 50 features to 500 features improves the motion estimation results. However, the relative improvement is less than 50% in all cases while the processing time increases by on order 600%. Obviously, diminishing returns dictates that the number of features should be kept as small as possible; in these example sequences, 50 features is sufficient.

For the 50 feature descent sequence and the linear motion estimation algorithm, the average translation error is 0.045 cm or 4.5% of the distance traveled. The average rotation error is 0.063 degrees from no rotation. These error values are similar to the expected motion errors (0.057 cm and 0.04 degrees) from Monte Carlo simulation given the parameters of the image sequence. The frame rate for this sequence is 4.01 Hz on a 174 Mhz R10000 SGI O^2.

For the 50 feature approach sequence and the linear motion estimation algorithm, the average translation error is 0.028 cm or 2.8% of the distance traveled. The average rotation error is 0.024 degrees for no rotation. These error values are similar to the expected motion errors (0.066 cm and 0.05 degrees) from Monte Carlo simulation given the parameters of the image sequence.The frame rate for this sequence is 2.91 Hz on a 174 Mhz R10000 SGI O^2.

The approach sequence results are more accurate because the resolution of the imager is greater which makes pixel track errors smaller. However, the approach sequence takes slightly longer to process because the larger image requires more time to detect features.

Motion estimation accuracy increases as feature disparity increases. To demonstrate this result, Figure 13 shows the expected motion estimation accuracy for the approach sequence given that features are tracked across multiple frames and the motion is estimated between the current frame and first frame. This experiment shows that for the approach sequence, as you increase the pixel disparity, the motion estimation becomes much better. In the last frame the median feature disparity is 23 pixels and the translational motion estimation accuracy is 0.004 cm. For a motion of 6 cm this is a 0.066% relative error. The rotational motion error is 0.003 degrees for no rotation.

In general, these results show that highly accurate motion estimation is possible using this algorithm. Furthermore, the processing times indicate that on-board image-based motion estimation is feasible. Given an order of magnitude difference in processing speed between our test computer (200 MIPS R10000) and a typical flight computer (20 MIPS RAD6000), each frame will take only a few seconds to process using the onboard CPU. This is more than enough speed for small body precision guidance and landing.

4 Performance Testing

Using Monte Carlo testing, the effect of sensor parameters (e.g., field of view, resolution), spacecraft trajectory (e.g., motion, altitude) and scene characteristics (e.g., surface scale) on the accuracy of body relative motion estimation can be determined empirically. We used these tests to search for the "best" sensor parameters for precise motion estimation and to predict the performance of the algorithm given a predetermined set of sensor parameters.

4.1 Monte Carlo Simulation

The procedure for a single Monte Carlo trial is as follows: First a synthetic surface is generated that represents the terrain of the small body within the field of view of the camera. An example of a synthetic surface is given in Figure 14. Next, a feature position in the first image is generated by randomly selecting a pixel in the image (feature position in first image). The 3-D position of the feature is found by intersecting its line of sight ray with the synthetic surface. Since the position of the camera for the second view is a known input, the 3-D point can be projected into the second view to determine its pixel position in the second image. Gaussian noise is then added to this feature pixel position to simulate feature tracking errors. This is repeated for however many features are requested. From these feature tracks, 5 DoF motion is estimated. Altimeter readings are computed by intersecting the line of sight for the altimeter (the camera optical axis) with the synthetic terrain, and computing distance between the sensor origin and the surface intersection. Gaussian noise is then added to the range value to simulate measurement noise in the altimeter. Using this altimeter reading, the complete 6 DoF motion is estimated.

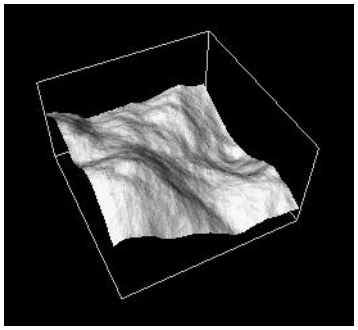


Figure 14: Example of a synthetic terrain used in Monte Carlo simulations.

The results of many Monte Carlo trials are used to determine empirically the motion estimation accuracy. For each trial, a new synthetic terrain is created so that dependencies on surface shape do not appear in the analysis. Since the ground truth motion is known, after each trial, the error in translational motion and the error in rotational motion can be computed. Statistics on these errors (average, median, standard deviation) for many trials constitute our estimates of motion accuracy.

This experiment assumes that features have been tracked robustly (no outliers) and accurately between frames. Feature tracking is excluded for two reasons: it depends on scene appearance which is too varied and difficult to parameterize; and previous studies have already modeled the accuracy of feature tracking, so its performance and accuracy are well understood [24].

For these tests some of the motion estimation parameters were fixed. Imager resolution was fixed at 1024 because this is the predicted resolution of low power space qualified imagers that will be available in the near future. Field of view was set to at 30 degrees because Monte Carlo simulations for variable motion and field of view show that 30 degrees is the best all around field of view that balances imager resolution, feature disparity and motion estimation accuracy [7]. The spacecraft altitude was set to 1000 m because this is a nominal altitude for the precision landing phase of the DS-4/Champollion mission. The altimeter range accuracy was set to 0.2 m following the specification given for the DS-4 Laser Radar Instrument. Feature tracking error was set at 0.17 pixels based on the analysis in [24]. The feature tracking disparity was set at 20 pixels because our experience has shown that it is reasonable to track features this far using multi-frame tracking. Scene surface scale is the absolute height variation between the closest and farthest terrain points in the field of view of the imager. The scene surface scale for generation of the terrain map was set to at 20% of altitude above the surface or 200 m. Finally, the number of tracks was set at 500 to enable highly accurate motion estimation without using an unreasonable number of features given that motion estimation time is always an issue.

The remaining parameters to investigate are the motion of the spacecraft and the scale estimation mode used in the algorithm.

4.2 Effect of Motion on Motion Accuracy

This investigation was performed to determine the effect of different spacecraft motions on motion estimation accuracies. To simplify this investigation, the space of possible motions was broken into two groups: descent (pure

translational motion) and pointing (pure rotational motion).

Descent can be parameterized by descent angle γ (See Figure 8), the angle between horizontal and the translation direction of the spacecraft. Given the above parameters, simulations showed that a translational motion accuracy of 0.22 m is expected independent of scale estimation mode and descent angle. At a fixed pixel disparity, the distance traveled between frames varies depending on the magnitude of translation. For a horizontal motion (γ=90˚), a 20 pixel disparity and 30˚field of view corresponds to a motion of 12 m. The motion error is then 0.22 m over 12 m or 1.8%. For a descent angle of γ=45˚ and a 30˚ field of view, a 20 pixel disparity corresponds to a motion of 17 m resulting in a motion error of 0.22 m over 17 m or 1.3%. Finally for vertical descent (γ=0˚)and a field of view of 30˚, a 20 pixel disparity corresponds to a 65 m motion. Thus the error is 0.22 m over 65 m or 0.34%.

By integrating this motion accuracy estimate from multiple frames as the spacecraft descends to the surface an upper bound on the expected horizontal landing position accuracy can be obtained. Simulations showed that the most accurate landing position occurs for the vertical descent with a 10 degree field of view. In this case the landing position accuracy is 3.6 meters. From a height of 1000 meters, this is an accuracy of 0.36% of the starting altitude.

To determine pointing accuracy we only investigated rotations with axes perpendicular to the camera Z-axis since rotations about the camera Z axis are unnecessary for pointing to surface targets. For a 30˚ field of view, a pixel disparity of 20 pixels corresponds to a rotation of 0.6˚ away from the optical axis. Simulations showed that given these parameters, a rotational motion estimation accuracy of 0.006 degrees or 1% of the rotational motion is expected.

4.3 Scale Estimation Mode

Descent angle and scene surface scale dictates which scale estimation mode to use during descent. Simulations were performed to determine at which descent angle the transition between scale estimation modes should occur. This angle is dependent on scene scale and is defined as the angle where translation magnitude errors of the two modes cross over.

The results of the simulation are shown in Figure 15. Inspection of the graph reveals that structure scale estimation should be used except when the surface is very flat (scale < 25 m at 1000 m altitude or 0.25% of altitude) or descent is very close to vertical (γ>88˚). Using this plot, it is possible to determine which scale estimation mode to use before scale estimation is performed. Descent angle is fully determined from 5 DoF image-based motion estimation. The scene scale can be determined before descent then though 3-D modeling or analysis of laser altimeter readings. Given this descent angle/scene scale data point, the scale estimation mode can be can be looked up using Figure 15. Figure 15 corresponds to a fixed set of imaging parameters, so if the imaging parameters change, a new plot will have to be generated.

5 Conclusion

We have developed and tested a software algorithm that enables onboard autonomous motion estimation near small bodies using descent camera imagery and laser altimetry. Through simulation and testing, we have shown that image-based motion estimation can decrease uncertainty in spacecraft motion to a level that makes landing on small, irregularly shaped, bodies feasible. Possible future work will include qualification of the algorithm as a flight experiment for the Deep Space 4/Champollion comet lander mission currently under study at the Jet Propulsion Laboratory. Current research is investigating the use of this algorithm to aid 3-D modeling of small bodies for terrain hazard assessment and comet absolute position estimation.

References

[1] A. Azarbayejani and A. Pentland. Recursive estimation of motion structure and focal length. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 6, pp. 562-575, June 1995.

[2] A. Benedetti and P. Perona. "Real-time 2-D feature detection on a reconfigurable computer." *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR'98)*, pp. 586-593, 1998.

[3] T. J. Broida, S. Chandrashekhar and R. Chellappa. Recursive 3-D motion estimation from a monocular image sequence. *IEEE Trans. Aerospace and Electronic Systems*, vol. 26, no. 4, pp. 639-656, July 1990.

[4] D. Brownlee, P. Tsou, K. Atkins, C. Yen, J. Vellinga, S. Price and B. Clark. Startdust: Finessing Expensive Cometary Sample Returns. *2nd Int'l Conf. Low-Cost Planetary Missions*, Laurel MD, IAA-L-0209, April 1996.

[5] T. D. Cole, A. F. Cheng, M. Zuber and D. Smith. The laser

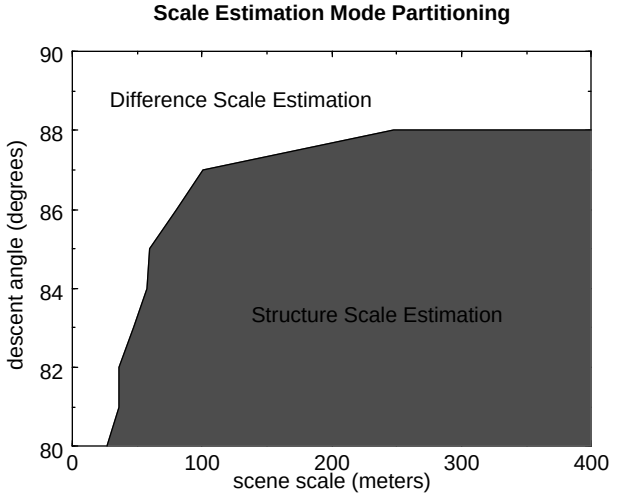


Figure 15: Scale Estimation Mode Partitioning from Monte Carlo Simulation.

rangefinder on the Near Earth Asteroid Rendezvous spacecraft. *2nd Int'l Conf. Low-Cost Planetary Missions*, Laurel MD, IAA-L-0910, April 1996.

[6] R. Hartley. "In Defence of the 8-point algorithm." *5th Int'l Conf. on Computer Vision (ICCV'95)*, pp. 1064-1070, 1995.

[7] A. Johnson. Monte Carlo experiments combining descent camera imagery and laser altimetry Web Reference: http://telerobotics.jpl.nasa.gov/people/johnson/ABLE/project/motion/monteCarlo.html, 1998.

[8] J. Kawaguchi, T. Uesugi, A Fujiwara and H. Matsuo. The Muses-C, world's first sample and return mission from a near earth asteroid Nereus. *2nd Int'l Conf. Low-Cost Planetary Missions*, Laurel MD, IAA-L-0202, April 1996.

[9] B. Leroy and G. Medioni. Crater detection for autonomous landing on asteroids. Submitted to *IEEE Computer Vision and Pattern Recognition*, 1999.

[10] H. Longuet-Higgins "A computer algorithm for reconstructing a scene from two projections." *Nature*, vol. 293, pp. 133-135, September 1981.

[11] L. Matthies. *Dynamic Stereo Vision.* Ph.D. Thesis, School of Computer Science, Carnegie MEllon University, 1989.

[12] J.K. Miller, B.G. Williams, W.E. Bollman, R.P. Davis, C.E. Helfrich, D.J.Scheeres, S.P. Synott, T.C. Wang, and D.K. Yeomans. "Navigation analysis for Eros rendezvous and orbital phases." *Journal Astronautical Sciences*, vol. 43, no. 4, pp. 453-476, 1995.

[13] S. Nozette et al. The Clementine mission to the Moon: Scientific Overview. *Science*, vol. 266, pp. 1835-1939, 1994.

[14] W. Press, S Teukolsky, W. Vetterling and B. Flannery. *Numerical Recipes in C, 2nd Edition.* Cambridge University Press, Cambridge, UK, 1992.

[15] J.E. Reidel, S. Bhaskaran, S.P. Synott, W.E. Bollman and G.W. Null. "An autonomous optical navigation and control system for interplanetary exploration missions." *2nd IAA Int'l Conf. on Low-Cost Planetary Missions*, Laurel MD, IAA-L-506, 1996.

[16] J. Shi and C. Tomasi. "Good Features to Track." *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR'94)*, pp. 593-600, 1994.

[17] D. E. Smith, M. T. Zuber, H. V. Frey, J. B. Garvin, J. W. Head, D. O. Muhleman, G. H. Pettengill, R. J. Phillips, S. C. Solomon, H. J. Zwally, W. B. Banerdt, and T. C. Duxbury. Topography of the northern hemisphere of Mars from the Mars Orbiter Laser Altimeter. *Science*, vol. 279, pp. 1686-1692, March 13th, 1998.

[18] R. Szeliski and S.B. Kang. Recovering 3-D shape and motion from image streams using non-linear least squares. *Journal Visual Communication and Image Representation*, vol. 5, no. 1, pp 10-28, March 1994.

[19] S. W. Thurman, C. D. Edwards, R. D. Kahn, A. Vijayaraghavan, R. C. Hastrup and R. J. Cesarone. Spacecraft navigation at Mars using earth-based and in situ radio tracking techniques. *World Space Congress 1992*, International Astronautical Federation, Washington DC, 1992.

[20] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal Robotics and Automation*, Vol. RA-3, No. 4, pp. 323-344, August 1987.

[21] .J. Weng, T. Huang and N. Ahuja. "Motion and structure from two perspective views: algorithms, error analysis and error estimation." *IEEE Pattern Analysis and Machine Intelligence*, vol 11, no. 5, pp. 451-476, 1989.

[22] J. Weng, N. Ahuja and T. Huang. "Optimal Motion and Structure Estimation." *IEEE Pattern Analysis and Machine Intelligence*, vol 15, no. 9, pp. 864-884, 1993.

[23] M.D. Wheeler and K. Ikeuchi. Iterative estimation of rotation and translation using the quaternion. Carnegie Mellon University School of Computer Science Technical Report CMU-CS-95-215, December 1995.

[24] Y. Xiong and L. Matthies. Error analysis of a Real-Time Stereo System. *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR'97)*, pp. 1087-1093, 1997.

[25] Z. Zhang. "Determining the epipolar geometry and its uncertainty: a review." *Int'l Jour. Computer Vision.* 1997.

Acknowledgments

We would like to thank Jean-Yves Bouguet for discussions on motion estimation. We would also like to thank Jackie Green for providing the comet analog.