AIAA SciTech Forum
7-11 January 2019, San Diego, California
AIAA Scitech 2019 Forum

10.2514/6.2019-1411

Vision-Based Navigation for the NASA Mars Helicopter

David S. Bayard[*] Dylan T. Conway[†] Roland Brockers[‡] Jeff Delaune[§] Larry Matthies[¶]

Håvard F. Grip[‖] Gene Merewether[**] Travis Brown[††] A. Miguel San Martin[‡‡]

Jet Propulsion Laboratory, California Institute of Technology, 91109

A small helicopter has recently been approved by NASA as an addition to the Mars 2020 rover mission. The helicopter will be deployed by the rover after landing on Mars, and operate independently thereafter. The main goal is to verify the feasibility of using helicopters for future Mars exploration missions through a series of fully autonomous flight demonstrations. In addition to the sophisticated dynamics and control functions needed to fly the helicopter in a thin Mars atmosphere, a key supporting function is the capability for autonomous navigation. Specifically, the navigation system must be reliable, fully self-contained, and operate without human intervention. This paper provides an overview of the Mars Helicopter navigation system, architecture, sensors, vision processing and state estimation algorithms. Special attention is given to the design choices to address unique constraints arising when flying autonomously on Mars. Flight test results indicate navigation performance is sufficient to support Mars flight operations.

I. Introduction

The use of helicopters promises to bridge a gap in current Mars exploration capabilities. Orbiters have provided high-altitude aerial imagery of Mars, but with limited resolution. Rovers provide rich and detailed imagery of the Martian surface, but move at a slow pace and are limited by traversability of the terrain and line-of-sight. Helicopters can quickly traverse large distances without being hindered by terrain, while providing detailed imagery of the surface from heights of a few meters to tens of meters above the surface. Paired with a rover, a helicopter can act as a scouting platform, helping to identify promising science targets or mapping the terrain ahead of the rover. Looking further ahead, helicopters may one day carry their own science payloads to areas that are inaccessible to rovers. An overview of the Mars Helicopter Technology Demonstrator is given in [2], and its guidance and control functions are discussed in [3][4]. The current paper will discuss the Mars Helicopter navigation system.

A CAD drawing of the Mars Helicopter is shown in Figure 1. The Mars Helicopter has two counter-rotating rotors that are 1.21 m in diameter. The vehicle stands approximately 80 cm in height, and weighs 1.8 kg. Compared to an Earth helicopter, the rotors are significantly oversized for its weight. This allows it to fly in the Martian atmosphere which has only 1-2% the density for that of Earth. The helicopter carries a payload as part of its fuselage, which is a cube-like structure containing the flight avionics, batteries, and sensors, all contained within a warm electronics box that is insulated and heated to protect against low night-time temperatures. The batteries are sized to provide energy for flights lasting over 90 s, while also supporting non-flight operations and night-time survival heating. A solar panel at the very top of the vehicle is used to charge batteries between flights.

[*]Mars Helicopter Navigation Lead, Guidance & Control Section, JPL, AIAA Associate Fellow, david.bayard@jpl.nasa.gov
[†]Guidance and Control Engineer, Guidance & Control Section, JPL, Dylan.T.Conway@jpl.nasa.gov
[‡]Robotics Technologist, Robotics Section, JPL, Roland.Brockers@jpl.nasa.gov
[§]Robotics Technologist, Robotics Section, JPL, Jeff.H.Delaune@jpl.nasa.gov
[¶]Senior Research Scientist, Robotics Section, JPL, lhm@jpl.nasa.gov
[‖]Mars Helicopter GNC & Aerodynamics Lead/Robotics Technologist, Guidance & Control Section, JPL, Havard.Grip@jpl.nasa.gov
[**]Robotics Technologist, Robotics Section, JPL, Gene.B.Merewether@jpl.nasa.gov
[††]Robotic Systems Engineer, Robotics Section, JPL, Travis.L.Brown@jpl.nasa.gov
[‡‡]Chief Engineer of Guidance & Control Section, JPL, miguel@jpl.nasa.gov

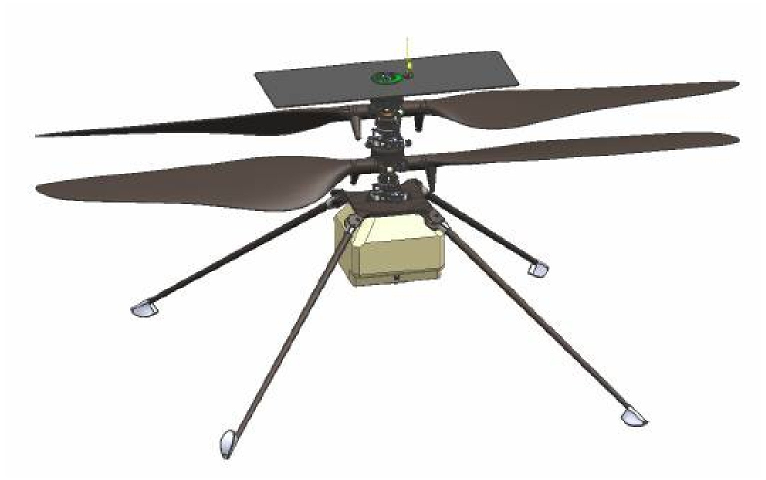


Figure 1. CAD drawing of the Mars Helicopter

II. Navigation Sensors

The navigation sensors are shown in Figure 2 mounted on the Electronic Control Module (ECM) which lies within the fuselage and constitutes the main component of the payload. The three main navigation sensors consist of a monochrome camera, an IMU, and a laser range-finder (LRF) which serves as an altimeter. An inclinometer is included for pre-flight calibration and initialization, but is not involved in the real-time navigation processing. A second IMU is included as backup, and a color camera is included for public outreach, but these sensors are outside the scope of this paper. As shown in Figure 2, the IMU and inclinometer are mounted together at the top of the ECM, while the camera and altimeter, are mounted together lower down and cantilevered off to the side. The camera and altimeter are nominally both aligned and nadir pointed.

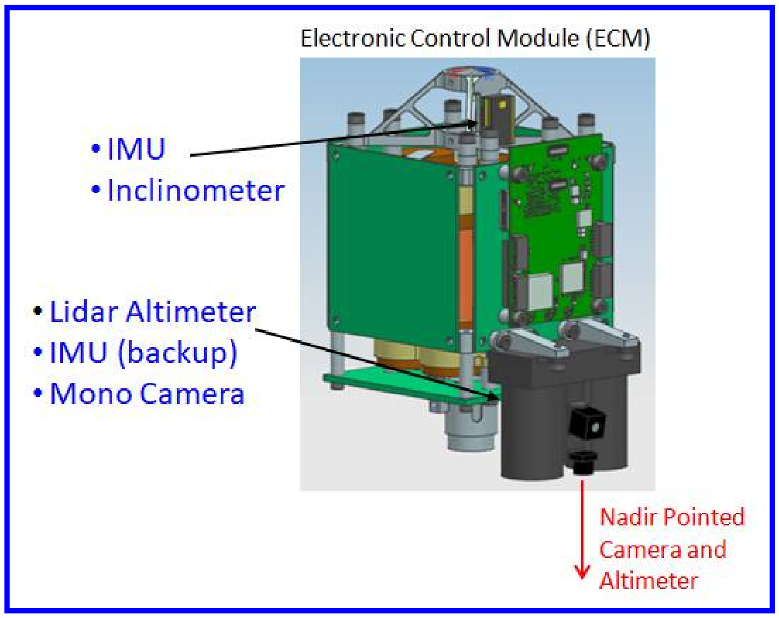

Electronic Control Module (ECM)
• IMU
• Inclinometer
• Lidar Altimeter
• IMU (backup)
• Mono Camera
Nadir Pointed Camera and Altimeter

Figure 2. Navigation sensors mounted on ECM

Specific navigation sensor units are listed below. These are all commercial off-the-shelf (COTS) miniature sensors, largely developed for the cell phone and lightweight drone markets.

- A Bosch Sensortech BMI-160 inertial measurement unit, for measuring 3-axis accelerations at 1600 Hz and angular rates at 3200 Hz
- A Garmin Lidar-Lite-V3 laser rangefinder (LRF), for measuring distance to the ground at 50 Hz
- A downward-looking 640×480 grayscale camera with an Omnivision OV7251 global-shutter sensor, providing images of the ground below the vehicle at 30 Hz
- A muRata SCA100T-D02 inclinometer, for measuring roll and pitch attitude prior to flight

III. Navigation Architecture

The navigation architecture is depicted in Figure 3. The architecture consists of a main flight computer (FC) a Nav Processor (NP) and an FPGA. The NP is a Snapdragon 801 SoC with four Krait 400 cores (COTS cell-phone processor). The FC is an ARM Cortex-R5 (COTS automotive-grade processor) which is 2x redundant for fault protection purposes. High-rate data (IMU, altimeter, inclinometer) is read into the FPGA which communicates synchronously with the FC and the sensor devices, and asynchronously with the NP. Camera images are read directly into the Nav Processor which uses cell-phone technology and can directly ingest video image sequences. The dedicated Nav Processor allows the CPU-intensive Vision Processing and State Estimation functions to be handled very efficiently using COTS cell phone technology. The Nav Processor serves to offload the FC and FPGA allowing them to more reliably handle important helicopter guidance and control functions. As shown in Figure 3, the Vision Processing and State Estimator functions are each assigned a full core of the quad-core Nav Processor.

The FPGA samples the IMU, the altimeter and inclinometer. The IMU is comprised of a gyro and accelerometer which are sampled at 3200 Hz and 1600 Hz, respectively. The IMU outputs in the body frame are first notch filtered at the first six rotor harmonic frequencies and then smoothed with a 90 Hz anti-aliasing filter. Data is then sent to the FC where coning and sculling corrections are applied at the high sampling rates, and then bias corrected (with a fixed initial gyro and accelerometer bias estimate only), and mapped to an inertial frame where it is integrated down to 500 Hz. During integration, the IMU data is time-aligned to a perfect 500 Hz grid which is needed because 500 Hz is not rationally commensurate with the original 1600 or 3200 Hz IMU data. The resulting "cleaned-up" 500 Hz delta-theta and delta-v IMU data and 50 Hz altimeter measurements are passed to the NP (via the FPGA) where they are ingested by the State Estimator function. IMU propagation is intentionally made redundant between the Flight Computer and Nav Processor in order to accommodate latencies and non-uniform packet arrival times associated with using asynchronous communications between the FPGA and the NP. Furthermore, the FC is architected to be a free-running integration of the IMU with updates that incorporate the latest bias and state information from the last good NP packet. With this approach, the system is robust to NP dropout and/or failure because the FC always continues to propagate the state from the last good navigation filter update.

It is worth noting that the helicopter avionics are largely comprised of commercial of-the-shelf (COTS) components. This is very different than most spacecraft payloads which require their avionics to be hardened in face of the harsh space environment and high radiation levels. While posing additional risk, this choice was essential for meeting tight mission cost, mass, and power constraints. Moreover, all components undergo vibration, thermal, and radiation tests to make sure that risks are consistent with a NASA Class D technology demonstration.

IV. Navigation Algorithm

A. MAVeN Approach

Many of the most reliable approaches to vision-based navigation make use of pre-mapped landmarks. This is because with a sufficient number of landmarks, the vehicle pose becomes fully observable from a single camera image. Unfortunately, using pre-mapped landmarks is not a practical option for the Mars Helicopter application because Mars terrain is observed predominantly from orbiting satellite assets, and is generally not mapped at sufficient resolution to support operations in close ground proximity.

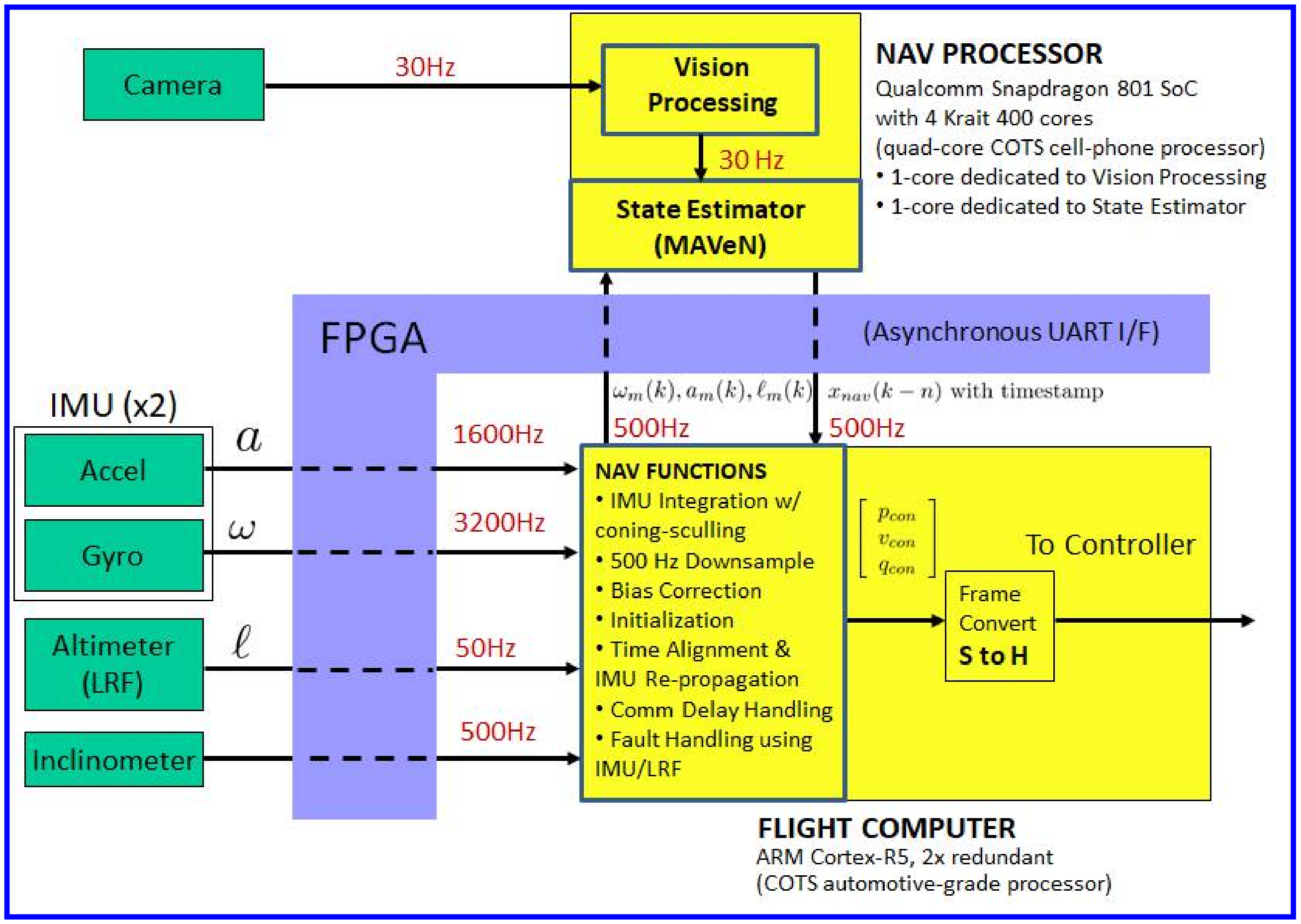

Camera
30Hz
Vision Processing
30 Hz
State Estimator (MAVeN)
NAV PROCESSOR
Qualcomm Snapdragon 801 SoC with 4 Krait 400 cores (quad-core COTS cell-phone processor)
• 1-core dedicated to Vision Processing
• 1-core dedicated to State Estimator
FPGA
(Asynchronous UART I/F)
$\omega_m(k), a_m(k), \ell_m(k)$
$x_{nav}(k-n)$ with timestamp
500Hz
500Hz
IMU (x2)
Accel
a
1600Hz
Gyro
ω
3200Hz
Altimeter (LRF)
ℓ
50Hz
Inclinometer
500Hz
NAV FUNCTIONS
• IMU Integration w/ coning-sculling
• 500 Hz Downsample
• Bias Correction
• Initialization
• Time Alignment & IMU Re-propagation
• Comm Delay Handling
• Fault Handling using IMU/LRF
p_{con} v_{con} q_{con}
To Controller
Frame Convert S to H
FLIGHT COMPUTER
ARM Cortex-R5, 2x redundant (COTS automotive-grade processor)

Figure 3. Navigation architecture

An alternative approach is to generate features on the fly and in real time as the vehicle maneuvers above the surface. Most generally, this problem can be addressed using SLAM (Simultaneous Localization and Mapping) algorithms from the literature [5][6]. While there has been some success applying SLAM to Earth-bound assets, full SLAM solutions are challenging for real-time space applications due to their over sized filter state dimensions. For example, SLAM augments the Kalman Filter with 3 states for each of the N features observed, so that retaining a memory of, say N=100 features, would require a filter state whose dimension exceeds 300. Such high-order filters are only marginally numerically stable and demand large amounts of on-board computation. While methods are being developed to perform management and real-time pruning of the number of features, such methods are relatively new and raise separate questions about reliability and adding complexity to the implementation.

The unavailability of mapped landmarks coupled with the challenges involved in flying a real-time high-order SLAM implementation, has motivated an alternative approach to Mars Helicopter navigation based on velocimetry. Here, the vision system is used to characterize relative motions of the vehicle from one image to the next, rather than to determine its absolute position. Providing good velocity information is also one of the main goals of the navigation system since it is essential for best supporting real-time control of a vehicle with complex dynamics. Of course a disadvantage of using velocimetry is that the navigation solution will drift with time since one is effectively integrating a ”noisy” velocity measurement. However this is offset by the short (approximately 90 second), flights for the Mars Helicopter, and the preference to implement simpler and lower-order navigation filters.

The velocimetry-based algorithm chosen to use for the Mars Helicopter application is MAVeN (minimal augmented state algorithm for vision-based navigation). MAVeN was originally developed as part of a JPL research project on comet exploration [8][9], and is adapted here for Mars Helicopter. MAVeN works similar to SLAM except that it avoids augmenting the state with feature vectors. MAVeN is mechanized as an Extended Kalman Filter (EKF) and has several advantages relative to other vision-based velocimetry methods, including a relatively low-order 21-state filter for full 6-DOF pose estimation, and the ability to maintain a stable hover condition. These advantages come at the cost of having to assume that a faceted shape model is available of the terrain being traversed. The availability and fidelity of such a faceted shape model will depend on the application. For example, applied to comet exploration, such a facet model would generally be available from remote spacecraft observations of the comet surface made prior to starting proximity operations. In contrast, for the Mars Helicopter application, a facet model or digital elevation map (DEM) is not available, so that the shape model must be taken simply as a single facet. While this facet could be defined as a tangent plane approximation to local terrain as informed by satellite data, it is instead just assumed to be flat (i.e. normal to local gravity) for simplicity. This assumption is justifed by flying over terrain that is expected to have sustained slopes of only 1-3 deg. An altimeter is added to provide scale, and to directly measure altitude above ground level (AGL) which is critical for helicopter guidance and to avoid ground collision.

B. Algorithm Description

The 21-state filter vector associated with MAVeN is given by,

$$x = \begin{bmatrix} p_S \\ v_S \\ q_S \\ -- \\ b_a \\ b_g \\ -- \\ p_B \\ q_B \end{bmatrix} \tag{1}$$

Here, p_S, v_S, q_S are the position, velocity and attitude quaternion states which comprise the Search state; b_a, b_g are bias terms for the accelerometer and gyro; and p_B, q_B are the position and attitude quaternion which comprise the Base state. Attitudes are represented using quaternions denoted using q which has 4 elements, of which only 3 unconstrained degrees-of-freedom are counted in the filter state size. Attitude is also represented equivalently using a DCOS matrix A, in which case the associated quaternion will be notated as $q(A)$. The Base state is a cloned version of the position and attitude part of the Search state, and is used for updating the EKF with information derived from simultaneously processing two images taken at different times. Cloned states are often added to vision-based methods help process relative measurement information [10][11][12]. As explained in [8] for the MAVeN algorithm, Base states are copied from Search states at the time instants t_B when Base images are taken (i.e., $p_B(t_B) = p_S(t_B)$, and $q_B(t_B) = q_S(t_B)$), and propagate with constant dynamics between Base images (i.e., $\dot{p}_B = 0, \dot{q}_B = 0$).

MAVeN's unique properties follow from the novel approach of projecting image features onto a shape model of the ground surface to use as pseudo-landmarks for the next image. This process is briefly sketched here with the use of Figure 4.

[1] Identify the first image as a Base image

[2] Use the current estimate of Base pose p_B, q_B to map features in the Base image onto the planar surface model e.g., f_1, f_2, f_3 in Figure 4. These feature positions will serve as pseudo-landmarks.

[3] Identify the next image as a Search image

[4] Match Search image features to the pseudo-landmarks mapped from most recent Base image. Assume that there are m matches.

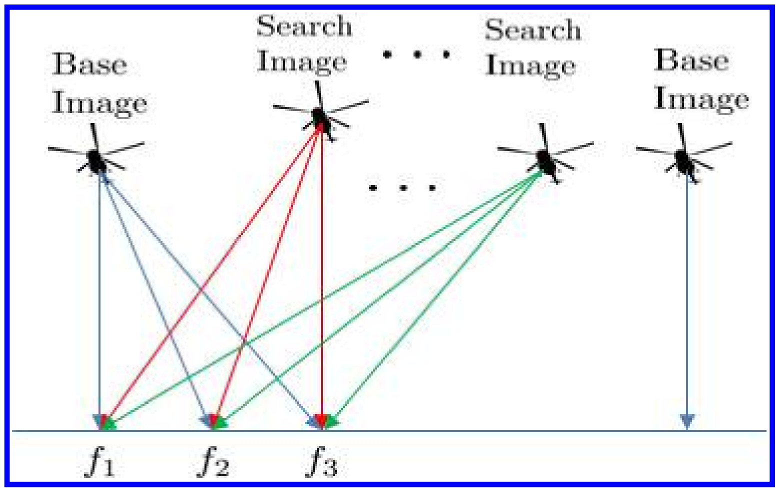


Figure 4. Base and Search frames

[5] Combine the m pseudo-landmark matches with current geometry to form a measurement that is a function of both the current Base and Search states,

$$y_i = h_i(p_S, q_S, p_B, q_B) + v_i, \quad i = 1..., m \tag{2}$$

Perform Kalman filter measurement and time updates.

[6] If the number of matched features drops below a threshold (or other relevant logic), declare the next image as a new Base image and go to [1]. Otherwise declare the next image as a Search image and go to [3]*

* The current mechanization is slightly more complicated where the new Base image in Figure 4 is also used simultaneously as a Search image associated with the previous Base frame. This intentional overlap minimizes the drift incurred between Base frames since it avoids a purely IMU-only period of integration.

REMARK 1 In a motionless hover condition, MAVeN is capable of sitting on the same Base image indefinitely which leads to very stable behavior. This property is generally not possible with non-SLAM approaches to velocimetry (cf., [14][15]).
REMARK 2 Updating the MAVeN EKF with consecutive Search images and altimeter updates automatically relocates the feature projections (e.g., f_1, f_2, f_3 in Figure 4), on the planar surface model to be consistent with the updated state estimate p_B, q_B, while holding bearing directions constant as observed in the previous Base image.

C. Stable Hover Property

The capability to hover is mission-critical for the Mars Helicopter. Unfortunately, performing state estimation during a motionless hover is very challenging for most vision-based velocimetry algorithms [14]. While algorithms based on the computationally intensive EKF-SLAM are generally able to hover [5][6], it is unfortunate that there are very few reduced-order estimators with this capability. This is because simpler estimators are often based on the essential matrix, the fundamental matrix, the 5-point algorithm [16], or they enforce epipolar or multistate constraints [15][24], and as such, achieve a singular condition that generates vacuous measurements under a motionless hover condition. To overcome this limitation, researchers have resorted to augmenting their filter state with persistent features [25][26] or invoking in-flight hover detection and switching to a different estimator in response [14]. While these modified methods can be made to work in practice, they complicate the implementation, make it harder to test, and still involve a large number of filter states. *In contrast, the MAVeN algorithm contains only 21 filter states and is able to achieve stable estimation during a completely motionless hover.* This capability follows directly from the novel approach of projecting image features onto a shape model of the ground surface to use as pseudo-landmarks. The stability of MAVeN during hover is demonstrated in Section VI, where a 200 second hover accumulates only 0.6 m of position error.

D. MAVeN Measurement Update

The main innovation of the MAVeN algorithm is the relative measurement update which is briefly outlined here. To help focus on the underlying geometry, camera measurement noise is temporarily dropped from discussion. The geometry of the Base and Search frames will be used as shown in Figure 5.

Four frames are used in the analysis: The Ground frame $\mathcal{F}_G$, the Camera frame $\mathcal{F}_C$, the Base frame $\mathcal{F}_B$ and the Search frame $\mathcal{F}_S$. The Ground frame $\mathcal{G}$ serves as the world frame for the scenario; the Base frame $\mathcal{F}_B$ is defined as coincident with the Camera frame $\mathcal{F}_C$ at the time instant t_B of the Base image (i.e., $\mathcal{F}_B = \mathcal{F}_C(t_B)$); and the Search frame $\mathcal{F}_S$ is defined as coincident with the Camera frame $\mathcal{F}_C$ at the time instant t_S of the Search image (i.e., $\mathcal{F}_S = \mathcal{F}_C(t_S)$). The DCOS matrix A_B maps a vector from $\mathcal{F}_G$ into $\mathcal{F}_B$, and the DCOS matrix A_S maps a vector from $\mathcal{F}_G$ into $\mathcal{F}_S$. These DCOS matrices are equivalently represented by the quaternions q_B and q_S, respectively in (1). The feature vector f_i lies in the ground plane and locates the i'th feature with respect to the origin of $\mathcal{F}_G$. Vector N is the unit normal to the ground plane.

For an arbitrary line-of-sight vector $r = [r_x, r_y, r_z]^T$ a pin-hole projection operator is defined as

$$\pi[r] = \begin{bmatrix} r_x/r_z \\ r_y/r_z \end{bmatrix} \tag{3}$$

Using this notation, a noiseless camera measurement $z = [u, v]$ of line-of-sight vector r can be written as,

$$z = \begin{bmatrix} u \\ v \end{bmatrix} = \pi(r) = \begin{bmatrix} r_x/r_z \\ r_y/r_z \end{bmatrix} \tag{4}$$

The corresponding unit direction vector d associated with the decomposition $r = d\|r\|$ can then be reconstructed from its noiseless measurement z as

$$d = \frac{\Pi(z)}{\|\Pi(z)\|} \tag{5}$$

where,

$$\Pi[z] = \begin{bmatrix} z \\ 1 \end{bmatrix} = \begin{bmatrix} \pi(r) \\ 1 \end{bmatrix} = \begin{bmatrix} r_x/r_z \\ r_y/r_z \\ 1 \end{bmatrix} \tag{6}$$

Based on this discussion, a noiseless camera measurement z of a line-of-sight vector r can be taken equivalently as its unit direction vector d in the decomposition $r = d\|r\|$. It will be convenient to apply this decomposition to the camera line-of-sight vectors r_{Bi} and r_{Si} which will be split into their magnitude and direction parts as $r_{Bi} = d_{Bi}\|r_{Bi}\|$ and $r_{Si} = d_{Si}\|r_{Si}\|$, respectively.

The various vectors shown in Figure 5 are assumed to be resolved as follows:

$p_B, p_S, f_i, N \in \mathcal{F}_G$

$r_{Bi}, d_{Bi} \in \mathcal{F}_B$

$r_{Si}, d_{Si} \in \mathcal{F}_S$

Forming a triangle at Base time t_B in Figure 5 and resolving all vectors in $\mathcal{F}_G$ gives

$$f_i = p_B + A_B^T r_B \tag{7}$$

Substituting $r_{Bi} = d_{Bi}\|r_{Bi}\|$ where d_{Bi} is a unit vector, gives

$$f_i = p_B + A_B^T d_{Bi}\|r_{Bi}\| \tag{8}$$

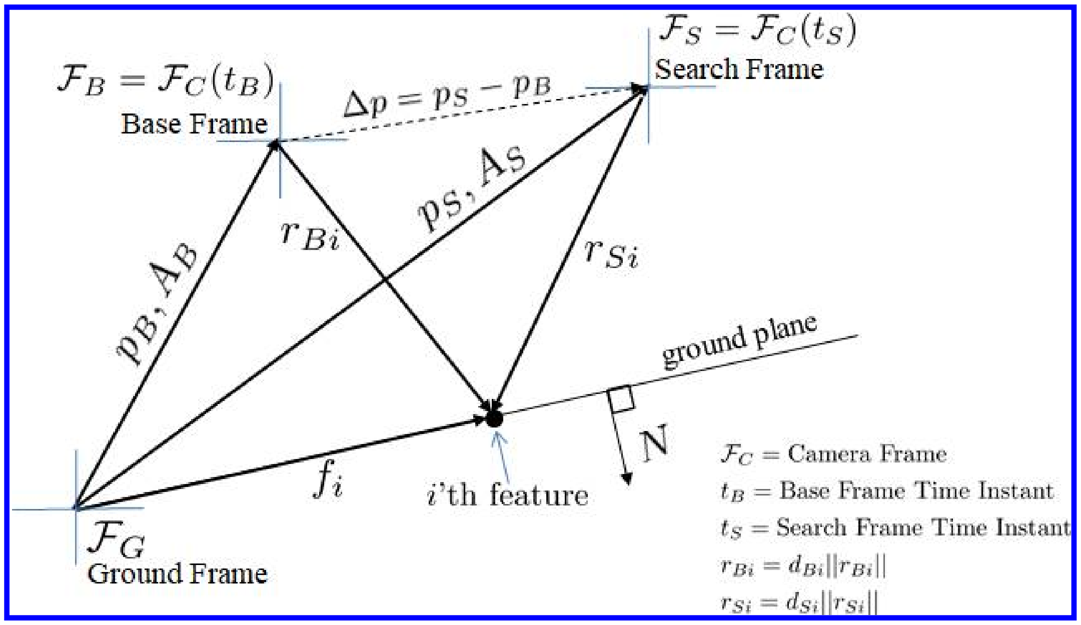

$\mathcal{F}_S = \mathcal{F}_C(t_S)$
Search Frame
$\mathcal{F}_B = \mathcal{F}_C(t_B)$
Base Frame
$\Delta p = p_S - p_B$
p_S, A_S
r_{Bi}
r_{Si}
p_B, A_B
ground plane
N
f_i
i'th feature
$\mathcal{F}_G$
Ground Frame
$\mathcal{F}_C$ = Camera Frame
t_B = Base Frame Time Instant
t_S = Search Frame Time Instant
$r_{Bi} = d_{Bi}\|r_{Bi}\|$
$r_{Si} = d_{Si}\|r_{Si}\|$

Figure 5. Geometry of Search and Base Frames

Taking the inner product of both sides with N gives

$$N^T f_i = N^T p_B + N^T A_B^T d_{Bi} \|r_{Bi}\| \tag{9}$$

Since N is normal to the ground plane in which f_i lies, it follows that $N^T f_i = 0$, and the above equation can be rearranged to solve for $\|r_{Bi}\|$ as

$$\|r_{Bi}\| = \frac{-N^T p_B}{N^T A_B^T d_{Bi}} \tag{10}$$

Substituting (10) into (8) gives

$$f_i = p_B - \frac{A_B^T d_{Bi} N^T}{N^T A_B^T d_{Bi}} p_B = \left(I - \frac{A_B^T d_{Bi} N^T}{N^T A_B^T d_{Bi}} \right) p_B \tag{11}$$

This is a useful formula showing that the feature vector f_i is a function of the cloned part of the state p_B, A_B, and the noiseless camera measurement d_{B_i} of the i'th feature taken at Base time t_B.

Consider the camera measurement update at Search time t_S. Forming a triangle at Search time t_S in Figure 5 and resolving all vectors in $\mathcal{F}_G$ gives

$$r_{Si} = A_S f_i - A_S p_S \tag{12}$$

Unfortunately, the vector f_i in this expression corresponds to an unmapped landmark and its coordinates are unknown. As such, expression (12) cannot be used directly to generate a standard mapped landmark type measurement update. Of course, one approach is to add f_i to the filter state which is an approach similar to SLAM. However, in the interest of keeping the filter dimension low, the main insight of MAVeN is to instead use the expression (11) for f_i derived from the earlier Base image. Specifically, substituting f_i from (11) into (12) gives,

$$r_{Si} = A_S \left(I - \frac{A_B^T d_{Bi} N^T}{N^T A_B^T d_{Bi}} \right) p_B - A_S p_S \tag{13}$$

It is worth noting that the formulation can progress from this point without any need to augment the state with f_i. Here, the formula (11) for f_i can be interpreted as defining pseudo-landmarks that can be used as if they were real landmarks in the navigation filter update. Of course care must now be taken to model f_i as a function of state, rather than simply a known vector, which creates additional complexity that is not required when having real mapped landmarks. Happily, the line-of-sight vector r_{Si} in (13) is now purely a function of state x through the quantities p_B, A_B, p_S, A_S, and a function of the exogenous quantity d_{Bi} which is assumed known from the earlier camera measurement at time t_B.

Applying the pin-hole projection operation (3) to the line-of-sight vector (13) gives a camera measurement of the form

$$z_i = \pi[r_{Si}] \triangleq h_i(p_S, A_S, p_B, A_B) \tag{14}$$

where,

$$h_i(p_S, A_s, p_B, A_B) = \pi\left[A_S\left(I - \frac{A_B^T d_{Bi} N^T}{N^T A_B^T d_{Bi}}\right)p_B - A_S p_S\right] \tag{15}$$

Since the only unknowns in this equation are p_S, A_s, p_B, A_B (where d_{Bi} is assumed known from the noiseless camera measurement at Base time t_B), and these quantities are part of the state vector x in (1), this equation has the form $z_i = h(x)$ which is consistent with standard nonlinear filtering formulations.

Until now we have assumed that there is no noise. To properly treat the underlying filtering problem, the camera noise needs to be added back into the formulation. At Search time t_S, the camera measurement z_i in (14) actually has the form

$$y_i = z_i + v_i \tag{16}$$

where y_i is a noisy measurement, and v_i is the measurement noise. Fortunately, the additive noise form (16)) remains consistent with standard nonlinear filtering formulations.

At time t_B the bearing measurement d_{B_i} will also be noisy. Since camera noise in d_{B_i} enters at Base time t_B, it becomes correlated with all subsequent Search frame measurement updates of the form (16). For simplicity, MAVeN assumes that the camera noise at each Base Frame is sufficiently small so that this correlation can be neglected. Rigorously dealing with the issue of correlated measurement noise remains as an area for future investigation. This small camera noise assumption at Base frame times, together with the planar ground assumption represent the only two approximations required in formulating MAVeN as a nonlinear filtering problem.

E. Lessons Learned

1. Vertical Channel Tuning

In the vertical channel, there is a difference between inertial altitude (as measured by inertial sensors such as an IMU or GPS), and altitude above ground level (AGL, as measured by the LRF and camera). These two quantities will be the same only when traversing perfectly flat and level ground. When traversing non-flat or irregular terrain, these two definitions of altitude conflict and can differ significantly. Without using GPS, and without using mapped landmarks or a digital elevation map (DEM), MAVeN will be unable to completely separate these two altitude definitions.

Nevertheless, the navigation filter is intentionally detuned to try to produce an AGL estimate of vertical position, while producing an inertial estimate of vertical velocity. This design is necessary because AGL height information is critical for avoiding collisions with the ground, while inertial velocity information is critical for properly controlling helicopter dynamics. Roughly speaking, filter tuning is performed by increasing LRF and camera weighting in the vertical position, and IMU weighting in the vertical velocity. Since detuning represents a compromise, the filter still generates noticeable systematic errors in vertical velocity estimates when flying over rough terrain. Fortunately, expected terrain for the Mars Helicopter mission is expected to be fairly flat. However for stress testing purposes, such systematic velocity errors are intentionally demonstrated in Section VI.

2. *Attitude States*

The original MAVeN algorithm design [8] only estimates translational states under the assumption that attitude is perfectly known. This is an appropriate assumption in mission scenarios where the spacecraft has a star tracker and high accuracy gyros. However, this mission does not have an absolute attitude sensor nor does it have navigation-grade gyros. Although the BMI-160 angle-random-walk spec of $7 \times 10^{-3}\ rad/s/\sqrt{Hz}$ would be sufficient to meet attitude knowledge requirements, early IMU testing revealed that the sensor is sensitive to vibration. The BMI-160 spec sheets lists a g-sensitivity of $0.1°/s/g$: a 1-g vibration level can induce a $0.1°/s$ gyro bias which would produce a $9°$ error over a 90 second flight. This is unacceptable which motivated the augmentation of the MAVeN filter state with attitude states. This extension of MAVeN is straight-forward to implement and extends the filter state dimension from 12 to 21 (three for current attitude, three for base attitude, and three for gyro bias). One drawback of this approach is that the attitude estimate becomes sensitive to non-flat terrain. For example, when flying forward over a long uphill stretch, the pitch error will converge to the slope of the hill. Nevertheless, the MAVeN attitude estimate is expected to be better than the gyro-only attitude estimate for the operational terrain which is expected to have sustained slopes no larger than 1-3 deg.

3. *Outlier Feature Suppression*

Visual feature tracking inevitably produces occasional outlier measurements (i.e. those with very large centroiding error). Outlier measurements can corrupt the filter state estimate if they are not down-weighted or discarded. One challenge in designing an algorithm for feature suppression is the possibility of *lockout*: the current state estimate is in error which causes good measurements to be incorrectly suppressed and therefore prevents a state update. The navigation system has a two-tiered approach to deal with this. The first tier is a base-to-search homography RANSAC algorithm to identify and discard outliers. This algorithm finds the largest set of base and search features that are consistent with a homography that maps base features to search features. By making the first tier independent of the navigation state, the lockout problem is avoided.

The first tier can fail if there are many outliers that are mutually consistent. This can happen when many features are detected on the shadow of the helicopter. To provide robustness to this failure mode, a second tier of outlier suppression is implemented inside of the MAVeN feature update. The magnitude of the innovation is used to assign a measurement weight on a per-feature basis. The weighting function is the Huber loss function [17] which assigns a weight between 0 and 1. A weight of 1 is assigned if the residual is less than a threshold. Beyond this threshold, the weight monotonically decreases. Down-weighting, as compared to a hard accept-reject rule, reduces the risk of lockout because a persistent update signal will still pass through to the estimated state.

V. Vision Processing

A. Overview

To provide the navigation filter with positions of selected feature points over a sequence of images, a three step process establishes the detection and tracking capability required by the MAVeN Base frame/Search frame procedure: Detection of features in a Base frame, tracking of features in subsequent frames, and an outlier rejection step to avoid feeding invalid feature tracks to the navigation filter. Note, that feature detection and tracking is performed on distorted images from the navigation camera, reducing the need for initial rectification of the input images.

B. Feature Detection

To provide the template-based feature tracker with a set of distinct pixel positions (visual features) that maximize the potential of being easily trackable in subsequent frames [18], we deploy a feature detection step at each Base frame to identify pixels with significant brightness changes in their local vicinity. To optimize performance and run-time of the feature detector, we use a modified FAST corner detector [19][20] to detect candidate feature points.

FAST explores the differences in brightness between an evaluated center pixel and neighboring pixels that are located on a circle around the center pixel. If neighboring pixels on a continuous arc around the center pixel (FAST arc) have consistently higher (or lower) brightness values than the center pixel, the center pixel is classified as a FAST feature. The detection result can be influenced by choosing a brightness threshold that defines the minimum brightness deviation between neighboring pixels and the center pixel, the radius of the circle in which neighboring pixels are evaluated, and the minimum required length of the FAST arc. Common values for these parameters are e.g. a radius of 5 pixels which yields a FAST window of 11x11 pixels and defines a circle around the center pixel containing 16 neighbors, and a minimum arc length of 9 pixels. Common FAST brightness thresholds - referred to as *FAST threshold* in the rest of the paper - are usually within the 5% range of the maximum brightness interval of the image.

FAST achieves a significant speed-up by initially evaluating only a few selected pixel positions on the neighbor arc to discard candidate features early on if the continuous arc condition is violated - eliminating the need to examine all neighbors for each pixel. While this accounts for significant speed-up in the majority of images, it introduces a run-time dependence on the texture content of the image. To limit the execution time of feature detection, we deploy a scanning scheme that uses a row stride length to go through the image during feature evaluation. Feature detection stops either when the whole image is scanned, or a maximum number of candidate features are found. This maximizes the distribution of feature candidates spread over the image, while bounding the maximum execution time.

To reduce the number of features around strong image brightness changes, we perform a non-maximum suppression step on all candidate features. Features are assigned a *FAST score* based on the brightness differences of the center pixel and the pixels on the FAST arc, and then tested for maximum FAST score in a 3x3 local neighborhood. If another feature with a higher score is located next to the tested feature, the tested feature is eliminated.

All surviving features pass through a final sorting step to only select the strongest features for ingestion into the navigation filter. To maximize the distribution of features over the image, the image is divided into a 3x3 grid of tiles, and features in each tile are sorted based on their FAST score. Finally, the n strongest features in each tile are accepted as Base frame features.

C. Feature Tracking

For each new image, we deploy a feature tracking step that matches feature positions in the previous frame to the new frame. This is done independently of Base frame generation to ensure continuous tracking.

To track features, we use a Kanade-Lucas-Tomasi (KLT) tracking framework. KLT uses an iterative gradient-descent search algorithm based on pixel differences in a local window (*template*) around a feature position [21] to estimate a new feature position in the current frame. To maximize robustness and minimize execution time, we deploy KLT on an image pyramid comparable to [22] with a fixed number of 3 levels and a template window size of 11x11 pixels. Additionally, feature positions are initialized prior to tracking through a derotation step, which integrates delta gyro measurements between frames to predict future feature locations through large rotations (see Section F below).

We further limit the maximum number of KLT iterations and discard any feature that did not converge during the iteration.

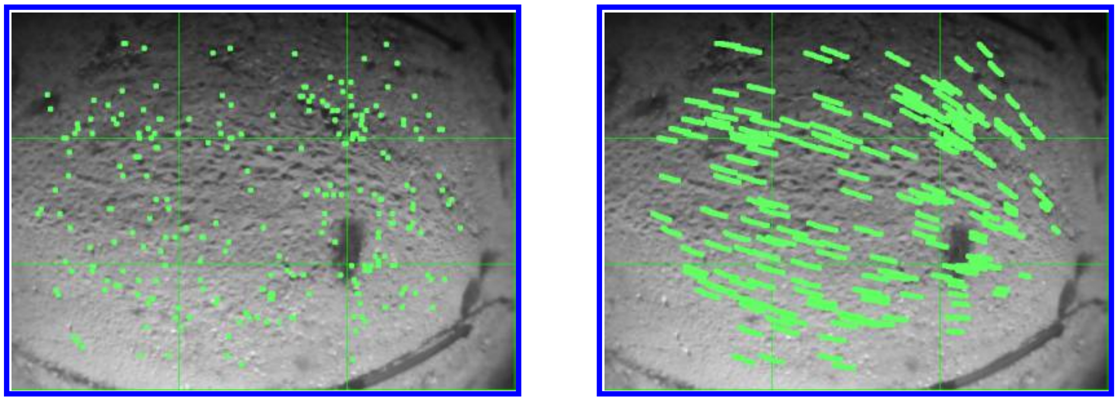


Figure 6. Feature tracking example. Left: A new Base frame is triggered and 252 features are detected with an equal distribution over a 3x3 image tile grid. Right: Green tracks show features that survived continuous tracking over 10 frames.

D. Outlier Rejection

Since KLT might get caught in a local minimum during its gradient-descend based search algorithm, a small number of false feature matches might survive the tracking step. To eliminate potential false matches, we apply an outlier rejection step to all newly matched features, that involves a homography-based RANSAC algorithm to identify the largest feature inlier set between the most recent Base frame and the current Search frame. Using a homography constraint serves the additional purpose of helping to enforce the ground plane assumption of the navigation filter. Specifically, only features that are located on a common ground plane between a Base frame and a Search frame survive the outlier rejection which naturally better conditions the selected features for the MAVeN state estimator. As an additional benefit, this scheme guarantees that all features that are detected on non-static texture (e.g., the heliocpter shadow), are also eliminated by the outlier rejection step under vehicle translation (but not pure rotation). Figure 6 shows an example of a Base frame and a subsequent Search frame with feature tracks that are continuously tracked over 10 frames.

E. When to Trigger New Base Frame

In support of the MAVeN navigation algorithm, the vision processing logic must determine when to establish a new Base frame. A new Base frame is required whenever the number of feature tracks drops below a certain threshold. In addition there is a desire to limit the maximum number of frames n between consecutive Base frames to help minimize drift induced from tracking over non-flat terrain. Specifically, since terrain tends to look locally planar over shorter spans, the navigation filter benefits from shortening the track lengths and becoming less sensitive to its planar ground assumption. The tracking algorithm evaluates the current tracking result and requests a new Base frame if the total number of tracked features after RANSAC drops below a prescribed number m, or if the number of image tiles that do not contain any features exceeds a maximum number k, whichever comes first.

In the current tuning, these numbers are chosen as $m = 40$ and $k = 3$. Additionally, track lengths are limited to $n = 10$ frames. Since images are processed at 30 Hz, this currently puts an upper bound of 1/3 sec on the time between consecutive Base frames. Generally, larger values for n best support hovering since there will be fewer Base-frame updates and therefore less drift, while lower values best support forward flight since shorter Base-to-Base intervals act to reduce navigation errors induced by traversing terrain that is not reasonably flat.

F. Tracking at High Angular Rates

The Mars Helicopter was designed to handle rotation rates of up to 80 deg/sec to help deal with wind gusts. Handling vehicle rates of 80 deg/sec presents a formidable challenge for feature tracking. This challenge was first addressed by speeding up the software to be able to process images at 30 Hz rate. However this alone was found to be insufficient for tracking features at 80 deg/sec while simultaneously ensuring a desired > 40 features per tracked frame. Additionally, it was found necessary to use gyro derotation. This is where delta-angles from the gyro are incorporated into the vision tracking algorithms. KLT linearizes the optical flow search around the initial feature position and becomes sensitive when the search position is initialized far from the true feature location. De-rotation is used to reduce the large initial displacements induced by vehicle rotation. Coupled with the high 30 Hz image frame rate, gyro-based derotation has been found to significantly reduce the chance of tracking outliers and makes the vision processing more robust.

G. Computational Constraints

In order to achieve the required image processing frame rate of 30 Hz, all image processing has to be executed within the 33 ms time limit between two successive frames. In a worst case scenario, the same image will need to be processed as both a Search frame and a Base frame. Specifically, this happens when feature tracking is performed on a Search frame and then a new Base frame is triggered because of insufficient feature tracks. This means that all components of the vision processing software combined have to fit within one frame period. Execution times for all algorithm steps on the target system are illustrated in Table 1. Results indicate a maximum execution time of 21.6 ms under this worst-case scnario, providing a reasonable margin against the 33 ms limit.

Component	min. [ms]	max. [ms]	avg. [ms]
Base Frame	4.3	8.8	4.7
Search Frame	5.9	12.8	6.6
total	10.2	21.6	11.3

Table 1. Runtimes on single Krait 400 core of Snapdragon 801 using well-textured images from simulation. Averages are calculated over 470 images.

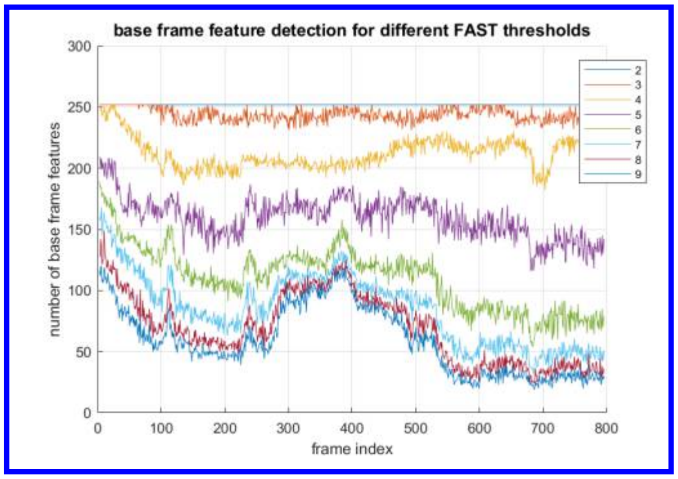
H. Lessons Learned

1. *FAST Threshold*

As described in Section B, the number of derived FAST features for a given image depends on the chosen FAST threshold that establishes a minimum brightness difference between a candidate feature and its neighboring pixels on the FAST arc. Features with large brightness changes usually mark unique image content, increasing the likelihood of being tracked in subsequent frames. But if the FAST threshold is chosen too high, fewer features are found and a minimum number of desired features might not be reached. This problem is commonly solved by applying the FAST detector to the same image with decreasing FAST thresholds, until the desired number of features is reached. Since this dramatically increases the number of runs through the image and the required execution time, we chose to set the FAST threshold to a low value and limit the number of extracted candidate features while applying the described stride scheme to ensure good feature distribution in the image. This has the advantage that only one run through the image is needed. Nevertheless, execution time is increased by the non-sequential memory access of the stride scheme, but we found that this effect is only marginal for images of VGA size and the capability of modern CPUs to store significant amounts of data in cache memory (the Snapdragon 801 has 2MB L2 cache on-chip).

Figure 7 illustrates the dependence of the number of detected features on the texture content of an image. Here, we apply the FAST detector with various score thresholds to a stress case environment of very low-textured terrain. As can be seen, a larger FAST threshold limits the number of features detected in the image.

To guarantee a sufficient amount of features in equivalent scenes, a FAST threshold of 3 was selected for our feature tracking frame work.



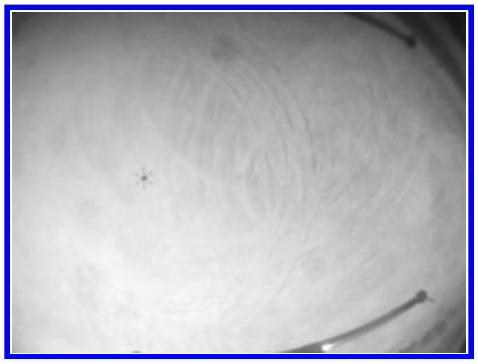


Figure 7. Influence of FAST threshold on the number of detected FAST features (left) using a navigation image from a low-textured scene over sand (right)

2. *COTS Hardware*

The Mars Helicopter would not have been possible without the low size, weight and power functionality offered by COTS hardware. The Snapdragon processor's four cores were assigned such that an entire core was dedicated to vision processing and an entire core to the navigation filter. This turned out to be essential for achieving the 30 Hz image frame rate needed to track features through wind-gust-induced vehicle rates of up to 80 deg/sec. There were however several major challenges that had to be overcome. Performance of the Snapdragon processor is temperature dependent, and it throttles back when it gets hot. Fortunately, thermal issues are less of a problem in the relatively short Mars Helicopter mission flights. Nevertheless, alternative approaches may be needed on future missions requiring long duration flights. Another problem is that the COTS Linus OS (Linaro) is not a real-time operating system, and its latencies are load-dependent and increase under heavy computation. Load-dependent latencies caused significant challenges for software repeatability and developing reliable real-time estimation and control functions. Finally, proprietary closed source software drivers for camera control and internal image processing made it difficult to change behavior of internal functions. Autoexposure, contrast settings, and internal image sharpening filters were all largely black box functions that had to be optimized using tedious manual cut-and-try methods based on dedicated flight tests.

3. *Autoexposure*

The OV7251 camera sensor requires a companion software autoexposure and autogain algorithm to cope with terrain having changing albedo, as it operates with manually adjusted *exposure time* and *analog gain* parameter setpoints. We use a vendor-supplied algorithm that adjusts both of these parameters based on image statistics in an attempt to optimize a data-driven function predicting image quality for feature tracking. The algorithm is tuned to reduce analog gain in order to minimize image noise, while relying on the expected radiance at the image sensor when viewing typical scenes on Mars to keep the exposure times low enough to eliminate motion blur.

In addition to adjusting the exposure time and analog gain as described above, the maximum allowable exposure *change* between subsequent frames is clamped to a specific value. The value is chosen such that the KLT tracker, which incorporates a constant brightness assumption at the core of the gradient-descent algorithm, is able to reliably track through exposure adjustments. In order to speed initial convergence of the autoexposure algorithm, the frame-to-frame clamp is removed during initialization of the navigation algorithm on the ground, and then reapplied during flight.

4. *Sharpening / Brightness / Contrast*

The Nav Processor carries out image preprocessing in dedicated on-chip hardware, the behavior of which can only be modified in limited ways and is tailored to cell-phone use cases. Although this hardware can be bypassed to deliver raw sensor data, we choose to enable this hardware to carry out radiometric darkening compensation, which improves the ability of the KLT tracker to track features from image center to edge and vice-versa.

This dedicated hardware also adjusts brightness and contrast of the images, and we tune this adjustment to enhance edge content in the images while avoiding pixel saturation. Specific values were chosen by capturing palettes of images on a static Mars-like scene with fixed illumination, and then quantifying FAST scores and ability of KLT to converge to the same pixel location being perturbed to varying degrees.

Unfortunately, the Nav Processor image processing hardware also has some undesirable characteristics. For example, a localized edge detection filter triggers adaptive blurring of the image in regions with low edge response, and spatially highpass filters regions with higher edge response. Although this is desirable for human viewing of commercial cell-phone images (because it has a sharpening effect that reduces image noise in regions with low spatial frequency content while enhancing sharpness in regions with higher edge content), this functionality degrades feature detection and tracking performance, especially when the edge content changes frame-to-frame. This sharpening filter was even sometimes found to cause undesirable frame-to-frame flickering in processed pixel intensities around edge content. Simple methods could not be found to turn it off. A binary routine was eventually obtained from Qualcomm that could turn it off, but it came too late to implement since all major testing and validation had already been performed.

VI. Flight Test Results

Flight testing was performed to evaluate expected performance by flying over regions local to JPL and the surrounding Arroyo area. The test platform was a commercial hexacopter outfitted with a payload consisting of an engineering model of the Mars Helicopter flight avionics and navigation sensors. Additional sensors were mounted on the hexacopter platform for ground truthing purposes, consisting of an RTK-GPS and a STIM300 IMU. The navigation filter was operated in real-time during these tests, although not actively used for feedback control of the vehicle.

Results are shown in this section for baseline flights representative of those to be flown as part of the Mars Helicopter mission scenarios, as well as additional stress test flights intentionally designed to challenge the vision processing and navigation filter capabilities. In all cases, the navigation filter performance is that of the 21 state MAVeN estimator operating without the benefit of mapped landmarks. Navigation requirements ensure stable helicopter control, and are given approximately as 3 m position error and 50 cm/sec velocity error, interpreted as the worst-case error norm over each flight, i.e., $\max_t \|e(t)\|$.

In all runs, ground truth position errors are fairly small at approximately 2-3 cm. However ground truth velocity errors are fairly large at 15-25 cm/s, and mostly due to high-frequency noise from back-differencing. As a result, velocity error performance numbers are fairly conservative.

A. Long Duration Hover Flight of 200 sec

Results from a long duration hover test are summarized in Figure 8. This test challenges the navigation filter to maintain a stable hover condition of 200 seconds in duration without accumulating significant drift. This is a factor of 2 longer than currently planned Mars Helicopter hovers which can be approximately 90 seconds duration.

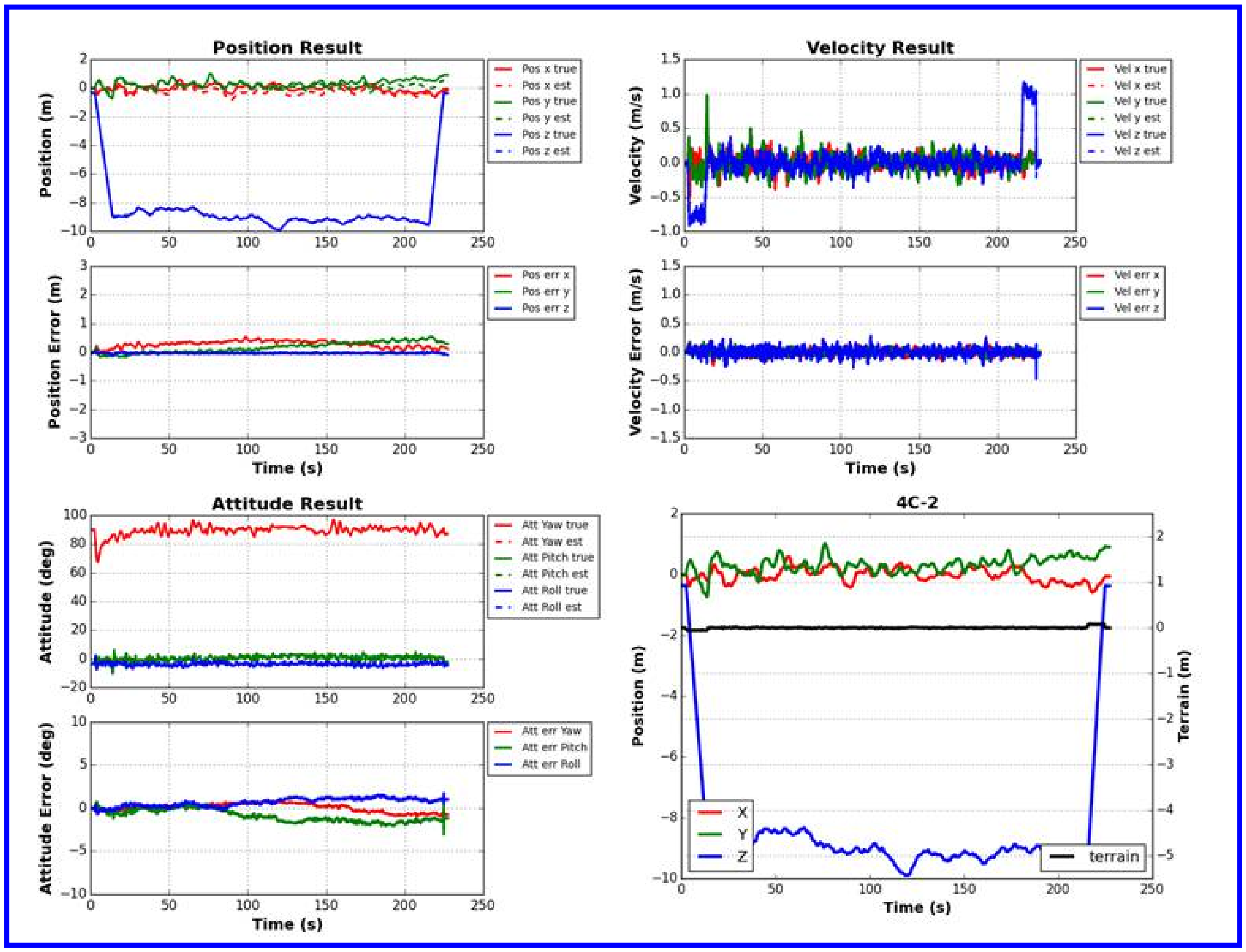


Figure 8. Hover flight for 200 sec

As expected, drifts accumulate with time over the 200 seconds. The worst-case error norms are given by 0.6 m in position and 32 cm/sec in velocity. The 32 cm/sec is dominated by ground truth errors as seen in the top right velocity plot of Figure 8. This performance provides a good demonstration of MAVeN's stable hover properties, and is consistent with accuracies needed to support the Mars Helicopter mission.

B. Forward Flight of 160m

Results from a forward flight test are summarized in Figure 9. This test challenges the navigation filter over a flight distance of 160 m and a flight duration of 120 sec. This distance is approximately as long as currently planned Mars Helicopter mission flights which can be up to 150 m. The terrain height profile (black trace) is plotted in the lower right plot of Figure 9, and its axis is shown on the right side of the plot. The terrain profile is symmetric because the flight traverses the same terrain going out and coming back. The slope is approximately 1 deg which is relatively flat and representative of the Mars mission.

Worst-case error norms are given as 1.22 m in position and 26 cm/sec in velocity. This performance is consistent with accuracies needed to support the Mars Helicopter mission.

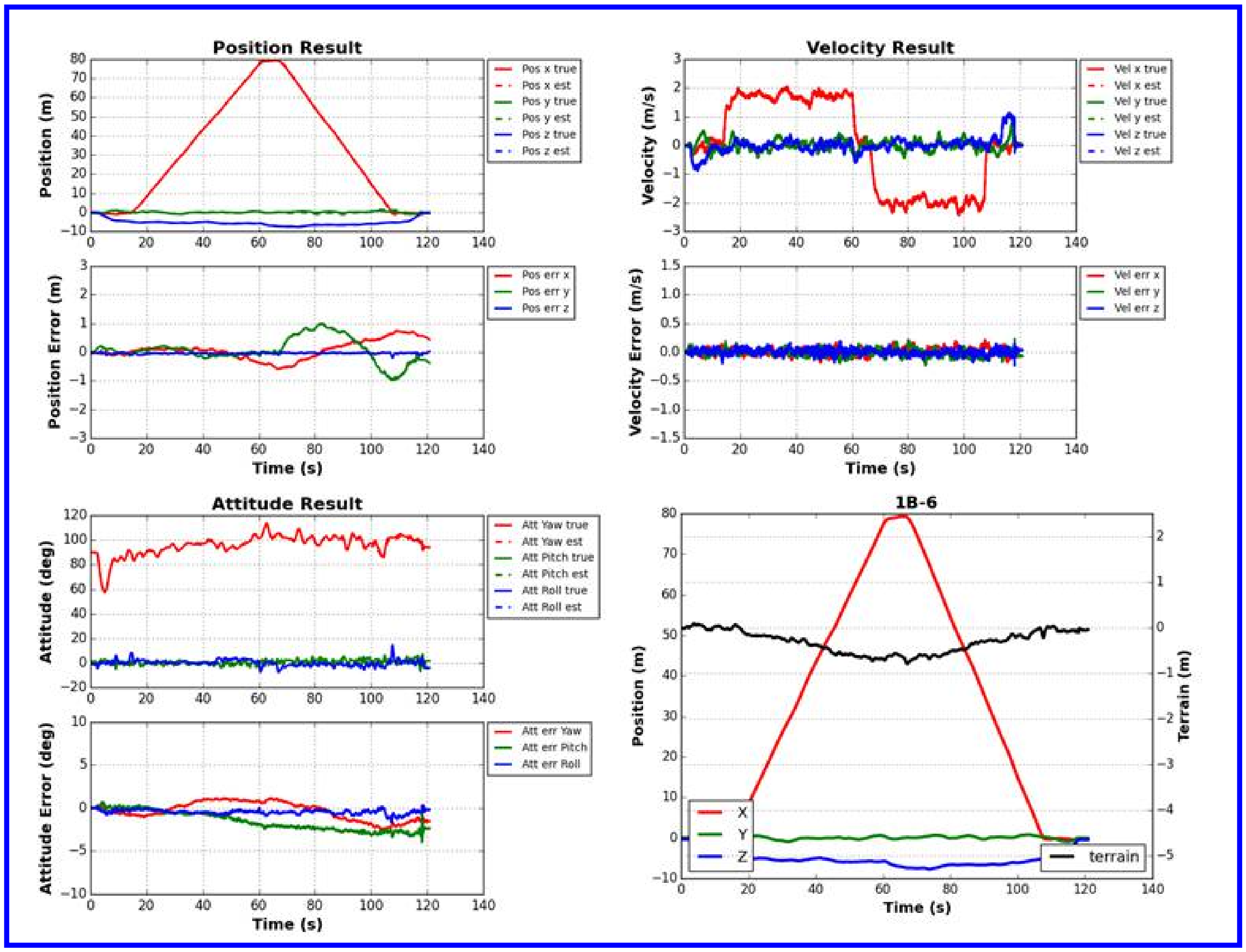


Figure 9. Forward flight traverse of 160 m

C. Forward Flight of 400m Over Rough Terrain

Results from a forward flight test over rough terrain are summarized in Figure 10. This test challenges the navigation filter over a flight distance of 400 m and a flight duration of 275 sec. This is a factor of 2.7 longer in distance and a factor of 3 longer in duration than for currently planned Mars Helicopter mission scenarios. As before, the terrain height profile (black trace) is plotted in the lower right plot of Figure 10. Besides the long distance and time duration, a main challenge is the rough terrain which varies by 2 meters in altitude as it is traversed, and contains sharp discontinuities. This represents a stress test of the navigation filter's flat ground plane assumption.

Worst-case errors are given by 2.51 m in position and 49 cm/sec in velocity. The maximum velocity errors (see blue traces in the top-right plot), are obtained at the time instants of the sharp discontinuities at 60 and 225 sec. Here, the altimeter picks up terrain discontinuities and passes them into the navigation filter where they are partially attributed to changes in the vehicle's inertial velocity. While the vertical channel of the navigation filter is detuned to minimize this effect, the tuning is not perfect because of conflicting requirements to estimate AGL position while estimating inertial velocity. Performance in this run is consistent with accuracies needed to support the Mars Helicopter mission.

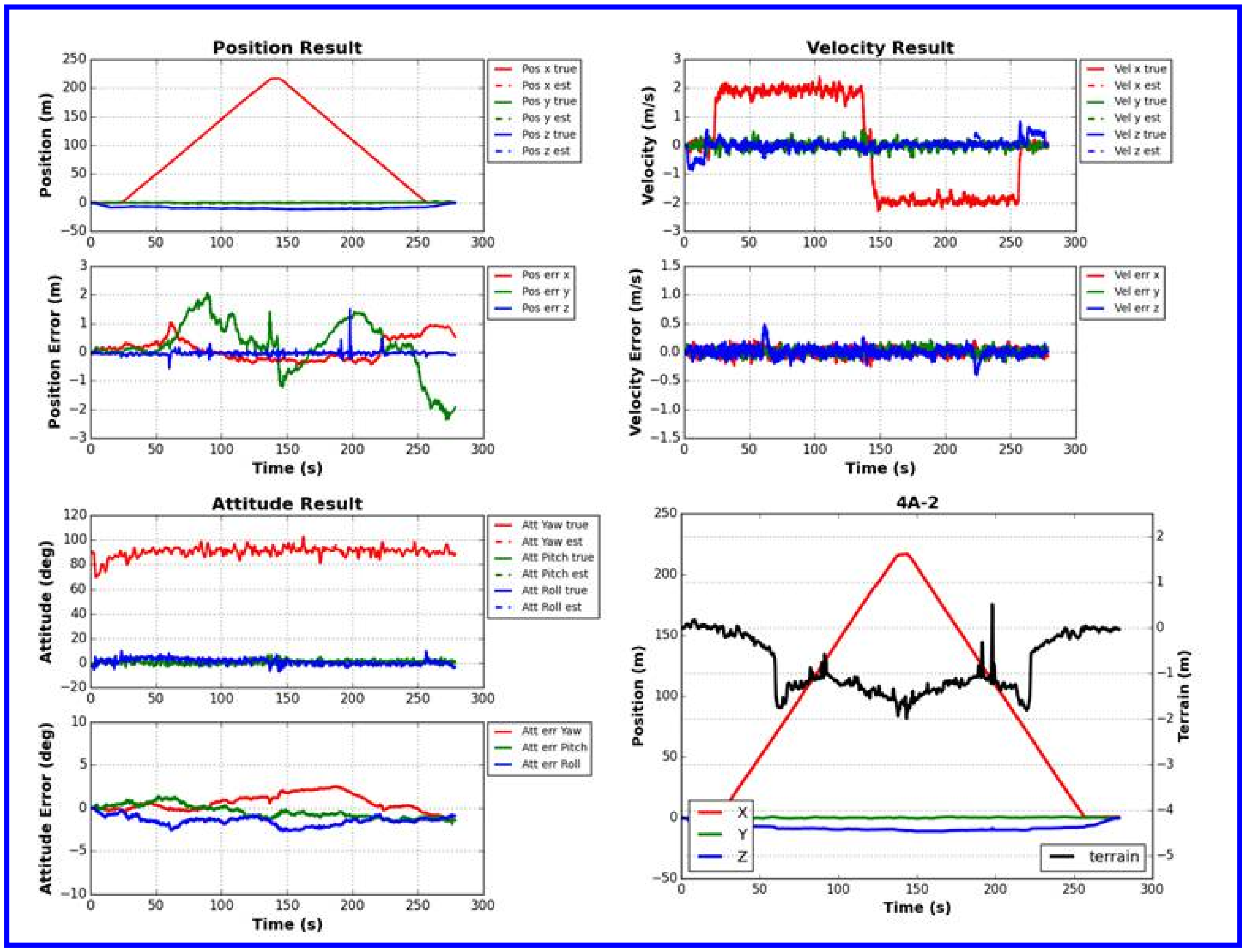


Figure 10. Forward flight traverse of 400 m over rough terrain

D. Forward Flight of 120 m Over Mounds

Mound flights represent some of the most stressful test flights because of violation of the flat ground plane assumption. Results from a forward flight of distance 120 m is shown in Figure 11. The main challenge is due to flying over significant mounds. Terrain height is given by the black trace in the lower right plot of Figure 11, and its axis is shown on the right side of the plot. It can be seen that the mounds grow by 1 meter and then dip by 3 meters in a short time span as they are traversed. This is a particularly challenging flight test since it represents a strong violation of MAVeN's planar ground assumption and a stress test of the filter's LRF altimeter detuning in the vertical channel.

Worst-case errors are given by 3.0 m in position and 68 cm/sec in velocity. The maximum velocity errors are obtained at the time instants in the vicinity of the mound peaks at 35 and 225 sec. Specifically, the mound position "pulses" are differentiated to become velocity error "doublets" (regions of alternating negative and positive peaks), that appear in the vertical channel velocity error plots (blue trace). Because the mounds are very broad (compare to Figure 10), and are traversed slowly, the navigation filter detuning discussed in Section E becomes less effective at preventing their bleeding into the vertical channel velocity estimates.

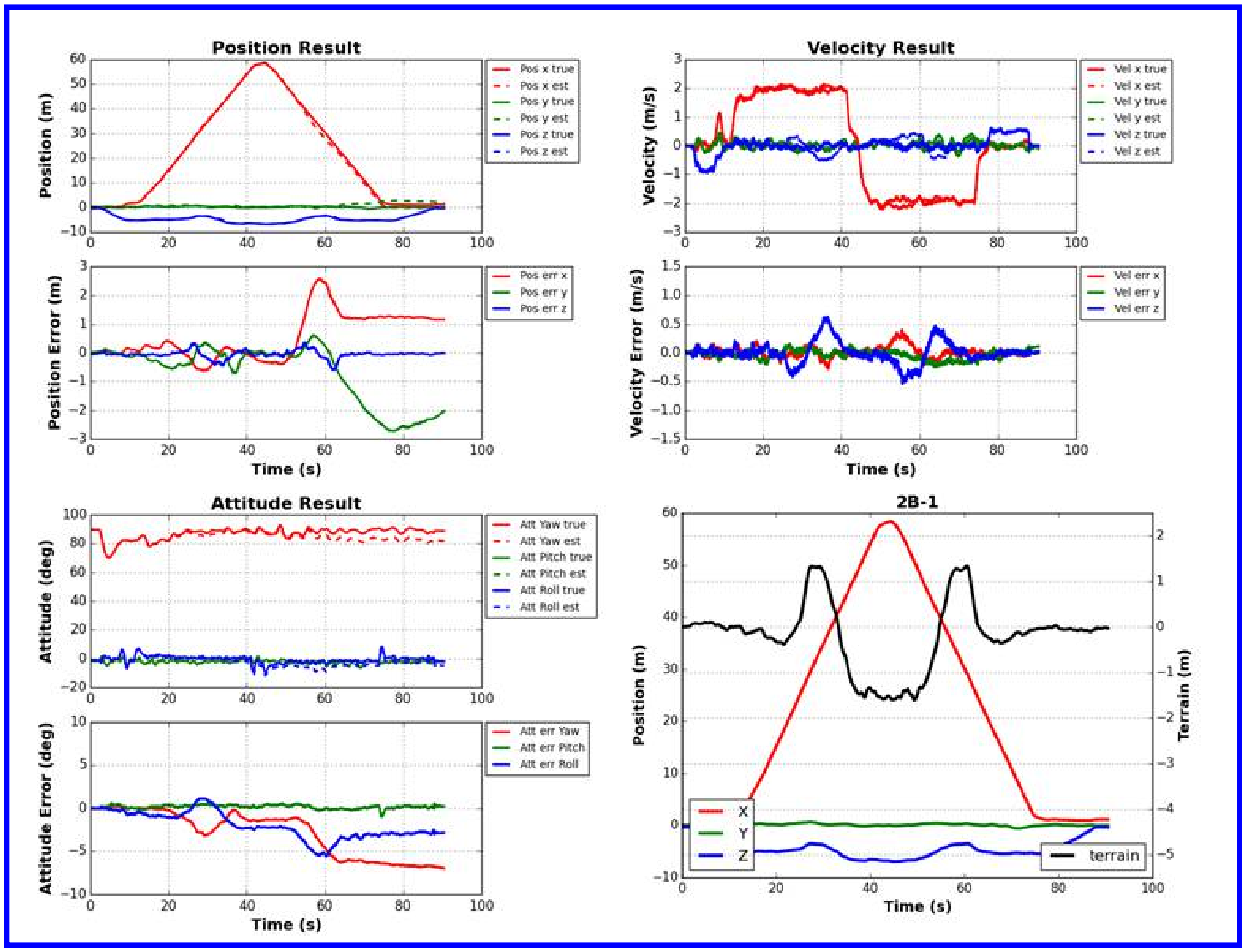


Figure 11. Forward flight traverse of 120 m over mounds

VII. Discussion

MAVeN only tracks features between the current Search image and the previous Base image. Because the Base frame is frequently reset as features are lost, MAVeN is effectively a long-baseline visual odometry algorithm: the relative position and attitude between the two images are measured, but not the absolute position and attitude. A typical consequence of visual odometry algorithms is that the absolute position and attitude error can grow over time. In the implemented navigation system, this is true for horizontal position and yaw. The LRF altimeter provides direct observability of vertical position which bounds vertical position error. In addition, the visual features and flat plane assumption provide observability of absolute pitch and roll when the vehicle is moving (but not when the vehicle is stationary). Although absolute pitch and roll are not directly measurable in hover flight, the growth in pitch and roll error is much slower than the gyro angle-random-walk rate. This is because rotational and translational motion are coupled in the IMU integration equations. The result is that pitch and roll error grow at a rate proportional (by the inverse of gravity) to the accelerometer bias growth rate.

A key advantage of MAVeN over other Simultaneous Localization and Mapping (SLAM) algorithms is that the state only needs to be augmented with six scalar elements - three for position and three for attitude. Other SLAM algorithms have a state augmentation scheme that scales with the number of features tracked [5][6]. The use of an LRF altimeter and an assumed ground plane enables MAVeN to estimate 3D position and velocity without introducing a scale ambiguity, resulting in reduced computational cost and improved numerical conditioning. Furthermore, unlike some loosely coupled architectures [13], MAVeN immediately benefits from feature data once it begins receiving images, without requiring a dedicated initialization step of the vision subsystem. Finally, unlike many other methods [14], MAVeN does not require vehicle motion to maintain observability during hover, which makes it ideal for hovering helicopter missions.

The two main disadvantages of MAVeN are sensitivity to rough terrain, due to the ground-plane assumption, and long-term drift in position and heading. For the Mars Helicopter technology demonstration, this is an acceptable tradeoff, because accuracy degradation is graceful and the algorithm has proven to be highly robust in both simulation and experiments.

VIII. Conclusions

A navigation system has been developed for the NASA Mars Helicopter. The design is driven by the need for completely autonomous operations and high reliability. The MAVeN navigation filter was deemed most useful because it has a relatively low filter order (21 states), and the unique property that it is able to hover stably. The novelty and intuition behind MAVeN's relative measurement update was outlined and discussed. Special vision-processing algorithms were discussed that address feature tracking over Mars-like terrain, under various stress conditions, and at high vehicle rotation rates. Also included are summaries of lessons learned from the development of the navigation system, and working with COTS hardware. Performance results indicate navigation accuracies of 1-3 m in position and 10-50 cm/sec in velocity over a flight envelope that includes flights having forward flight velocities of 1-5 m/s, hover durations of 200 sec, and 400 meters total distance traversed. These results are consistent with accuracies needed to support the Mars Helicopter mission.

Acknowledgments

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. This paper makes use of flight test data products acquired under a separate JPL internally funded research & technology development program (joint with CalTech), focused on studying the applicability of COTS hardware for future space missions [27]. The authors would like to thank all members of the COTS study team who implemented the ECM3 avionics flight test article and helped with its flight testing. In addition to authors, the JPL team included Theodore Tzanetos, Fernando Mier-Hicks, Gerik Kubiak, Lucas Leach, Russell Smith, and the CalTech team included Prof. Soon-Jo Chung, Thayjes Srivas, Amarbold Batzorig, Kai Matsuka, Alexei Harvard, Reza Nemovi, and Elaine Lowinger.

References

[1] Klein, G., and Murray, D., “Parallel tracking and mapping for small AR workspaces,” 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR 2007, pp. 225234, 2007.

[2] Bob Balaram, Timothy Canham, Courtney Duncan, Havard F. Grip, Wayne Johnson, Justin Maki, Amelia Quon, Ryan Stern, and David Zhu, “Mars Helicopter Technology Demonstrator,” 2018 AIAA Atmospheric Flight Mechanics Conference, p. 0023. 2018.

[3] Havard F. Grip, Daniel P. Scharf, Carlos Malpica, Wayne Johnson, Milan Mandic, Gurkirpal Singh, and Larry A. Young, “Guidance and control for a Mars Helicopter,” 2018 AIAA Guidance, Navigation, and Control Conference, p. 1849. 2018.

[4] Havard F. Grip, Johnny N. Lam, David S. Bayard, Dylan T. Conway, Gurkirpal Singh, Roland Brockers, Jeff Delaune, Larry Matthies, Carlos Malpica, Travis Brown, Abhindandan Jain, Miguel San Martin, and Gene Merewether, “Flight control system for NASA’s Mars Helicopter,” Proc. AIAA Scitech Conference, 2019.

[5] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse, “MonoSLAM: Real-time single camera SLAM,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 29, No. 6, 2007.

[6] Montiel, J. M., Civera, J., and Davison, A. J., “Unified inverse depth parametrization for monocular SLAM,” Robotics: Science and Systems, 2006.

[7] R. Smith, M. Self, and P. Cheeseman, “Estimating uncertain spatial relationships in robotics,” arXiv preprint arXiv:1304.3111, 2013.

[8] A. Miguel San Martin, David S. Bayard, Dylan T. Conway, Milan Mandic, Erik S. Bailey. “A Minimal State Augmentation Algorithm for Vision-Based Navigation without Using Mapped Landmarks,” 10th International ESA Conference on Guidance, Navigation & Control Systems, GNC 2017, Salzburg, Austria, May 29 - 2 June 2, 2017.

[9] A. Miguel San Martin, David S. Bayard, Dylan T. Conway, Milan Mandic, Erik S. Bailey, “A Minimal State Augmentation Algorithm for Vision-Based Navigation Without Using Mapped Landmarks (MAVeN),” New Technology Report, NTR # 50296, Software Copyright of Invention NPO 50296-CP, October 21, 2016.

[10] S. I. Roumeliotis and J. W. Burdick, “Stochastic cloning: A generalized framework for processing relative state measurements,” Proc. IEEE Int. Conf. Robot. Autom., Washington, DC, pp. 1788-1795, 2002.

[11] D.S. Bayard and P.B. Brugarolas, “An On-Board Vision-Based Spacecraft Estimation Algorithm for Small Body Exploration,” IEEE Transactions on Aerospace and Electronic Systems., Vol. 44, No. 1, pp. 243-260, January 2008.

[12] D.S. Bayard, “Reduced-Order Kalman Filtering with Relative Measurements,” AIAA J. Guidance, Control and Dynamics, Vol. 32, No. 2, pp. 679-686, March-April, 2009.

[13] Klein, G., and Murray, D., “Parallel tracking and mapping for small AR workspaces,” 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR 2007, pp. 225234, 2007.

[14] D.G. Kottas, K.J. Wu, and S.I. Roumeliotis, “Detecting and dealing with hovering maneuvers in vision-aided inertial navigation systems,” Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference, pp. 3172-3179, 2013.

[15] A.I. Mourikis and S.I. Roumeliotis, “A multi-state constraint Kalman filter for vision-aided inertial navigation,” Robotics and Automation, 2007 IEEE International Conference, pp. 3565-3572, 2007.

[16] D. Nister, “An efficient solution to the five-point relative pose problem,” IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 26, No. 6, June 2004.

[17] P.J. Huber “Robust estimation of a location parameter,” Annals of Mathematical Statistics, Vol 35, No. 1, pp. 73-101, 1964.

[18] Jianbo Shi and Carlo Tomasi, “Good features to track,” Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn., pages 593-600, 1994.

[19] E. Rosten, R. Porter, and T. Drummond, ”Faster and better: A machine learning approach to corner detection,” IEEE Trans. Pattern Analysis and Machine Learning, Vol. 32, No. 1, January 2010.

[20] E. Rosten and T. Drummond. ”Machine learning for high-speed corner detection,” European Conference on Computer Vision, pp. 430-443. Springer, Berlin, Heidelberg, 2006.

[21] B. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” Proceedings of the International Joint Conference on Artificial Intelligence, pages 674-679, 1981.

[22] Jean-Yves Bouguet, ”Pyramidal implementation of the affine Lucas Kanade feature tracker description of the algorithm,” Intel Corporation 5, no. 1-10, pp. 4, 2001.

[23] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* (2nd ed.). Cambridge University Press, 2003.

[24] S. Weiss, M.W. Achtelik, S. Lynen, M. Chli and R. Siegwart, “Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments,” Robotics and Automation (ICRA), 2012 IEEE International Conference, 957-964, 2012.

[25] M. Li and A.I. Mourikis “Vision-aided inertial navigation for resource-constrained systems,” Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference, pp. 1057-1063, 2012.

[26] B. Williams, N. Hudson, B. Tweddle, R. Brockers and L. Matthies, “Feature and pose constrained visual aided inertial navigation for computationally constrained aerial vehicles,” pp. 431-438, 2011.

[27] D. S. Bayard, D. Conway, R. Brockers, J. Delaune, H. Grip, F. Mier-Hicks, G. Kubiak, G. Merewether, L. Leach, R. Smith, L. Matthies, and T. Tzanetos, “6-DOF visual inertial navigation technology on SWAP constrained COTS platforms: Hexacopter Project Final Report,” Tech. Rep., NASA JPL, October 18, 2018.

This article has been cited by:

1. Håvard F. Grip, Johnny Lam, David S. Bayard, Dylan T. Conway, Gurkirpal Singh, Roland Brockers, Jeff H. Delaune, Larry H. Matthies, Carlos Malpica, Travis L. Brown, Abhinandan Jain, Alejandro M. San Martin, Gene B. Merewether. Flight Control System for NASA's Mars Helicopter . [Citation] [PDF] [PDF Plus]
2. Rémi Girard, Sébastien Mavromatis, Jean Sequeira, Nicolas Belanger, Guillaume Anoufa. A Vision-Based Assistance Key Differenciator for Helicopters Automonous Scalable Missions 202-210. [Crossref]