AUTONOMOUS SERENDIPITOUS SCIENCE ACQUISITION FOR PLANETS **(ASSAP)**

A. Aljabri, D. Eldred, R. Goddard, V. Gor, T. Kia, M. Rokey, D. Scheeres, P. Wolff

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, *California 91109*

Introduction

This paper discusses work in progress to develop and demonstrate a set of autonomy technologies that provide the onboard capability to redirect science observations. The onboard software modules will demonstrate a proof-of-concept autonomous task planning, sequencing, execution, and recovery from failures. Domain specific modules for navigation, maneuver implementation, command generation, and science observation redirection are the major functions chosen for onboard automation. These modules are developed in the context of a Mars orbiter mapping mission.

Key to meeting the objective is the integration of these modules into an end-to-end demonstration showing how the orbiter mission segment is autonomously conducted. The high-level mission functions consist of routine science operations, navigation, delta-v maneuvers, science redirection, momentum dumping and fault responses.

Reference Mission

To provide a realistic environment in which to prototype autonomous capabilities, a reference mission and spacecraft (Fig. 1) have been defined, called Autonomous Serendipitous Science Acquisition for Planets (ASSAP). The science goal of ASSAP is to return high resolution radar images of selected sites of geological interest on Mars. This mission will use an L-band radar to image Mars globally with SAR and then autonomously extract science data and select images for transmission back to Earth. For selected areas, optical and super-resolved radar images would be made for transmission back.

The spacecraft will autonomously determine when it is in the correct observing geometry for each of the super-resolution target sites, and at those times, autonomously maneuver into a ground-synchronous orbit which repeats the ground-track over the target site. After completion of imaging of one site, the spacecraft will autonomously maneuver back to a mapping orbit until another of the selected super-resolution sites comes into view.

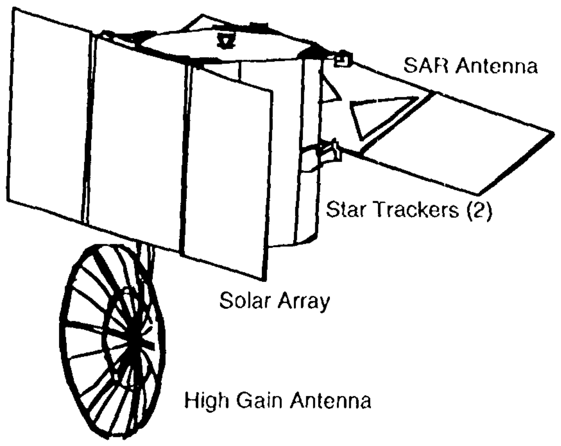

Figure 1 ASSAP Spacecraft

To illustrate how such a mission would operate, a day in the life of ASSAP is presented. Figure 2 shows the activities occurring on orbit 1200, an orbit during the southern hemisphere mapping phase. The spacecraft orbits the planet in a clockwise fashion. As the spacecraft approaches the equator, the SAR is powered up, initial radar values are set, and the spacecraft is maneuvered to an attitude corresponding to the start of mapping. From the equator to the southern pole, the radar collects data and ships it to a special purpose computer, which correlates the data in real-time. A second special purpose computer examines the correlated radar data, looking for features of interest. Figure 2 shows feature extraction occurring from the start of mapping until the crossing of the equator on the ascending side of the orbit.

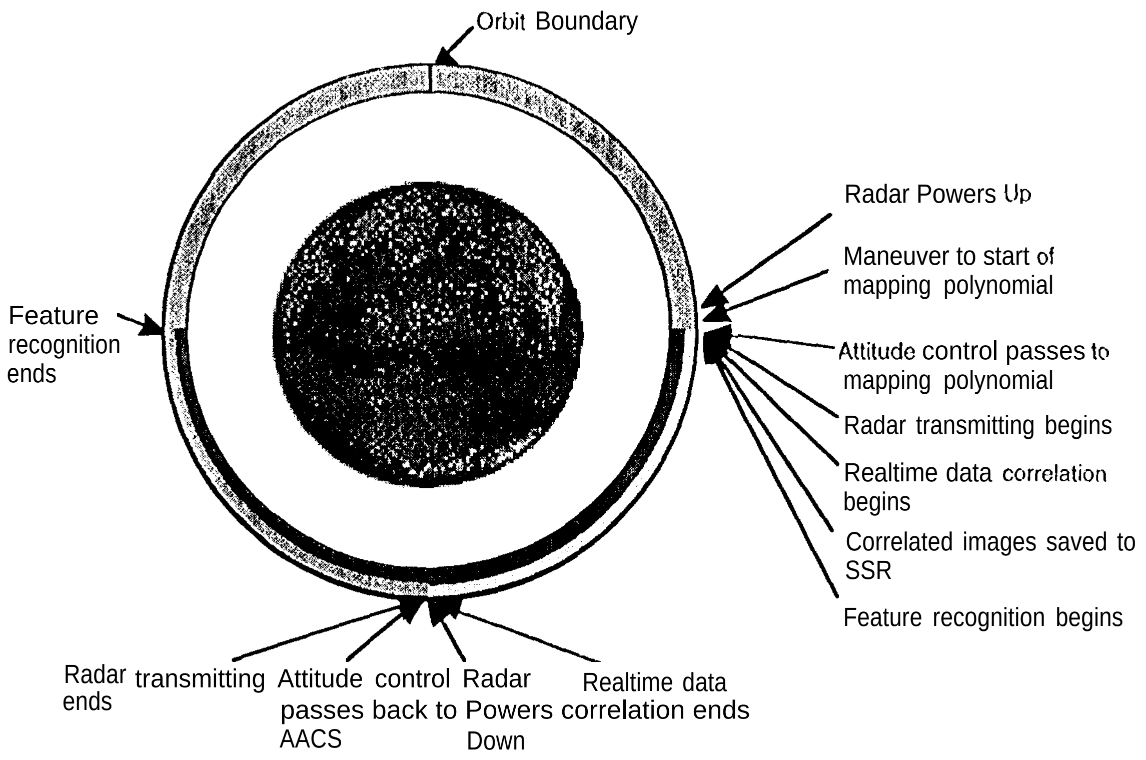
Orbit Boundary
Radar Powers Up
Maneuver to start of mapping polynomial
Attitude control passes to mapping polynomial
Radar transmitting begins
Realtime data correlation begins
Correlated images saved to SSR
Feature recognition begins
Feature recognition ends
Radar transmitting ends
Attitude control passes back to AACS
Radar Powers Down
Realtime data correlation ends

Figure 2 Orbit 1200 - Nominal Mapping

During the data collection period of orbit 1200, a request is made to freeze the orbit so that multiple images of a target can be taken, The request to freeze the orbit in this repeat pass configuration is executed on Orbit 1201.

Once the orbit is frozen, orbits follow a repeat pass collection profile which emphasizes targeted data collection. After the requisite number of repeat pass orbits are taken ASSAP autonomously maneuvers itself back into the standard data collection mode orbit.

ASSAP **Navigation Module**

ASSAP will demonstrate an integration of the science observation scheduling process with the navigation system. The scope of the current delivery is restricted and concentrates on computing and delivering requested ΔV's, spacecraft position and velocity information, and science observation times. The ASSAP trajectory modeling and propagation capabilities capture the functionality provided by the DPTRAJ (Double Precision Trajectory) software of the traditional ground based navigation system. Although the ASSAP navigation module does not provide the sheer breadth of force modeling options available in the large ground based legacy system, the limited set of physical models needed for ASSAP were largely inherited from the original software to facilitate timely implementation and to ensure accuracy and reliability. The computation of ΔV'S for science retargeting captures the functionality provided by the MOPS (Maneuver Operations Software) component of the ground based system. The amount of inherited code for the retargeting portions of the software was less than that for the trajectory force modeling because the dynamic and adaptive requirements for ASSAP are not well captured in' the existing MOPS system. Although the ASSAP navigation modules provide a large portion of the necessary capabilities currently available only by the ground based software, several aspects of a complete navigation system were omitted from the initial delivery.

Among the elements not delivered are

- Orbit determination capability
- Maintenance of a landmark data base
- Scheduling of optical navigation image frames

- Identification and location of known landmarks in an optical navigation image along with requisite pointing refinement (center finding)

These items are not needed for the demonstration of autonomous science observation and scheduling. It shall be assumed that the above elements can be made available in a future on-board navigation system.

Scheduling science observations

The navigation module will receive a list of features of interest for super-resolution imaging. The location will be assigned one or more of 30 ground track bins. Based upon the current orbit knowledge, the times for entry into each of the bins is obtained. This knowledge is passed back to the science module which chooses the next bin it wants the orbiter to enter. This bin id is passed back to the navigation module which then schedules maneuvers to enter and exit the 13 orbit repeat ground track bin. In addition, the landmark flyover times are recomputed using the latest trajectory information to support scheduling of the SAR observations. The navigation modules output will be

- Time of AV for entry into the 13 orbit repeat ground track
- Magnitude and direction of ΔV in inertial coordinates
- 1. ist of flyover times for feature(s) of interest
- Time of AV for exit of 13 orbit repeat ground track
- Magnitude and direction of exit AV

Ground track for SAR

The latitude and longitude search space shall be divided into 30 ground track “bins”. Each bin represents a 13 orbit repeat ground track reflecting the field of view of the SAR instrument. The corresponding node crossings of the different bins are offset in increments of 2p/(13 x 30) radians from each other. Thus the union of all 30 bins provides complete coverage of the planet. This section documents the assumptions and equations used to describe the ground track for a single bin. For the purpose of computing a ground track we assume an orbit of zero eccentricity and constant rate for the longitude of the ascending node. The node rate is designed to make the orbit sun synchronous. The longitude of the ascending node in the body-fixed frame is

$$\Omega = \Omega_0 + \dot{\Omega} t - \omega t$$

where

Ω_0 = longitude of first node crossing

Q = mean node rate

ω = rotation rate of Mars

t = time relative to first node crossing

We define the three unit vectors

$$\mathbf{r}_\Omega = \cos \Omega \hat{\mathbf{x}} + \sin \Omega \hat{\mathbf{y}}$$

$$\mathbf{r}_T = -\sin \Omega \cos i \hat{\mathbf{x}} + \cos \Omega \cos i \hat{\mathbf{y}} + \sin i \hat{\mathbf{z}}$$

$$\mathbf{r}_N = \sin \Omega \sin i \hat{\mathbf{x}} - \cos \Omega \sin i \hat{\mathbf{y}} + \cos i \hat{\mathbf{z}}$$

The spacecraft position can be expressed as

$$\mathbf{r} = a[\cos v \mathbf{r}_\Omega + \sin v \mathbf{r}_T]$$

where

a = semi-major axis (circular orbit radius)

i = inclination

v = true anomaly

Given a look angle b, the look direction is

$$\hat{\mathbf{r}}_{SAR} = - \cos \beta \hat{\mathbf{r}} + \sin \beta \mathbf{r}_N$$

The vector representing the location where the view point intersects the surface of Mars can be represented as

$$\mathbf{r}_{SAR} = \mathbf{r} + \alpha \hat{\mathbf{r}}_{SAR}$$

Solving for a so that the norm of $\mathbf{r}_{SAR}$ is equal to the planet radius yields the following expression for the location of the viewpoint on the surface of Mars

$$\mathbf{r}_{SAR} = a\left[\sin^2\beta + \cos\beta\sqrt{\left(\frac{r_0}{r}\right)^2 - \sin^2\beta}\right]\hat{\mathbf{r}} - a\sin\beta\left[\cos\beta - \sqrt{\left(\frac{r_0}{r}\right)^2 - \sin^2\beta}\right]\hat{\mathbf{r}}_N$$

where

r. = radius of Mars

The latitude and longitude can be readily computed from the components of $\mathbf{r}_{SAR}$.

To determine if and when a candidate location is visible given the location of a feature of interest (λ^*, δ^*), search for the time t* when the angular distance between the feature of interest and r_{SAR} is minimized. Given the field of view for the SAR instrument $\Delta\beta$ we evaluate the location of the aim point for a look angle of $\beta \pm ½\Delta\beta$ and compute the angular distance with respect to r_{SAR}. If the angular distance is be in the field of view of the instrument and t^* is the predicted flyover time. Otherwise, the feature is categorized to be outside the field of view covered by the current bin.greater than that between r_{SAR} and the feature of interest, then the feature is said to

ASSAP Command Generation Module

The command generation module for ASSAP, also called the auto-sequencer, is modeled on the process used by ground controllers to generate commands. In classic JPL missions, the process of commanding has followed the process outlined in Figure 3.

Requests, or “directives”, are given to the flight team which are then used to construct a profile for the time period in question. Profile construction is done by the sequence team, by hand on several missions (such as Voyager and Mars Observer) and with software on others (MSP on Magellan and Mirage on Galileo). A profile is then specified in terms of blocks, which are scripts of commands, which accomplish the specified directives. The SEQGEN program translates blocks into spacecraft commands and the SEQTRAN program translates commands into encoded bits, ready for transmission to the spacecraft, Figure 4 shows the advancement in commanding represented by the Mars Pathfinder project. VxWorks, a real-time operating system, now provides the capability to move the last step in the command generation process, namely translation of commands into bits and associated memory management, onboard the spacecraft itself.

Figure 5 shows the approach under study by ASSAP, where commanding is moved back to the directive level, with all subsequent command generation being done onboard the spacecraft. The method of this command generation is quite similar to the process on the ground, and is shown in high-level pseudo-code in Figure 6. The real-time operating system runs a process which acts as a command issue mechanism (the CDS of today), as well as a sequence generation process, a navigation process, and a guidance and control process, provides the foundation from which commands can be built in an autonomous fashion,

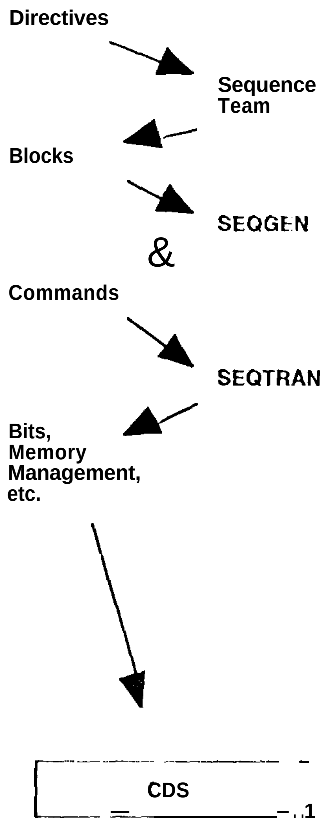

Figure 3 Current Command Generation Process

Directives, issued from the ground or from systems onboard ASSAP, are collected converted into a commanding profile. The profile is then turned into fully instantiated blocks, with parameter values being determined by information from the AutoNav and AutoAACS subsystems. Blocks are then expanded into commands, which are written to a file in the CDS and the CDS command process is notified a “new sequence has arrived.” It is important to note that in this approach, the block definitions

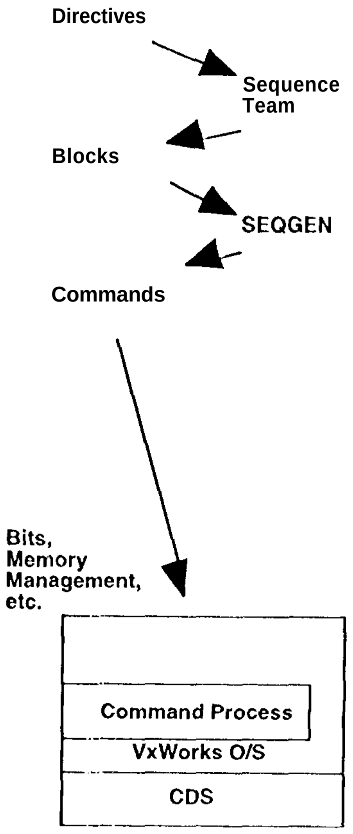

Figure 4 Pathfinder Command Generation Process

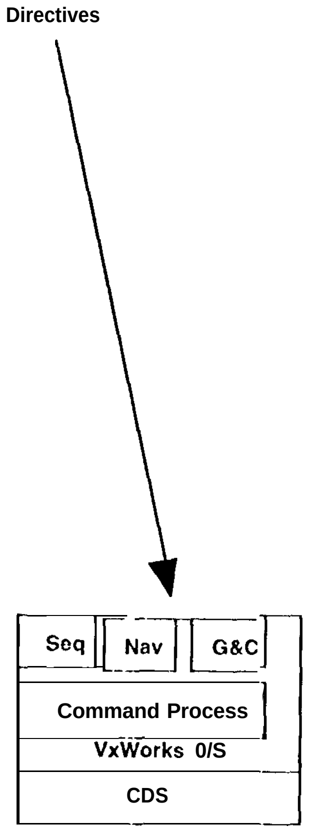

Figure 5 ASSAP Command Generation Process

and command definitions used by the ground staff are identical to those used by the autosequencer. Further, from the point of view of the CDS, sequences look identical, regardless if they are generated onboard on or the ground.

If such a system is implemented then we have the ability to implement autonomy in a way which naturally allows ground interaction, as the steps in command generation are essentially the same for both ground-based and autonomous generation. Since the steps and the tokens (blocks and commands) are the same, we have the ability to enable and disable autonomy, with easy transitions in and out. Also since both ground-based commanding and autonomous commanding are using the same structures, integration and test is greatly simplified. Finally, this method provides a simple conceptual model, well understood by operations personnel.

```
Loop begin
        housekeeping;
        when current orbit is executing do
                collect and evaluate directives;
                instantiate orbit profile;
                collect block parameters, communicating with autoGN&C;
                expand blocks into commands;
                write sequence file;
                notify CDS a new sequence has arrived;
End_loop;
```

Figure 6 High-Level Command Generation Description

ASSAP Science Recognition Module

The Science Recognition Module will demonstrate capability for adaptive autonomous science processing and data recognition and acquisition. It will also integrate with onboard S/C architecture and other software modules such as autonomous sequencing and autonomous GN&C.

Autonomous science object recognition allows transmission of data of interest only. This enables 2-3 orders of magnitude reduction in down link bandwidth, As a result the science return of the mission per dollar invested is maximized. Adaptive object recognition provides flexibility to the scientist to redefine the mission real-time. Autonomous and adaptive object recognizes enable reduction in S/W design and development cost. Derivation of super-resolved images permits usage of low resolution instruments (cheap, light and small instruments), therefore reducing the cost of launch and S/C design.

JARtool (JPL Adaptive Recognition tool) capabilities will be used as a baseline for producing trainable and adaptive recognizes in near real-time. Scientists indicate targets of interest by pointing at examples in the available imagery. JARtool automatically learns features and produces recognizes which are hard coded or uploaded into the spacecraft. Recognizes detect targets of interest in the newly collected data, and areas of interest only are down linked back to Earth. Automatically or on the scientists requests the data is examined in super-resolution mode. ARC super-resolution software will be used to enable recognition of small and/or faint objects.

The science unit interfaces with the rest of the system are shown in Figure 7. The science unit has four modes of operation: load recognizes mode, browse mode, feature extraction mode, and super-resolution mode. During load recognizes mode the new recognizes are uploaded into the system. The recognizes are usually 7 filters (as will be discussed later) and Gaussian statistics. In browse mode the image is fetched from the 'image location' in new image buffer, then the image is compressed to fit browse image description, followed by placing the image in downlink buffer and updating buffer pointers. While in feature extraction mode, the image is once again fetched from 'image location' of new image buffer, then the features of interest to the scientists are extracted from the image and placed into the downlink buffer, followed by buffer pointer's update. If the number of interesting features is large enough the request for detailed observation is generated by the system and placed in observation request buffer. The request for detailed observation is equivalent to the request for super-resolution. The difference is that super-resolution requests buffer holds requests generated by scientists on Earth, and observation requests buffer holds requests generated by science module. Super-resolution mode gets multiple images from super-resolution image buffer and puts those images in downlink buffer while updating buffer pointers.

The system produces three different types of output corresponding to last three modes of operation. Browse image is produced during browse mode, science features with the possible request for detailed observation (or super-resolution) are produced by feature extraction mode, and super-resolution images are produced by super-resolution mode. Output type specifies the type of output, necessary for data interpretation. The data stream is terminated by the 'end of data flag'.

The algorithm consists of two parts, a training part and a production part. The training part is done on Earth. H consists of labeling and learning. First the scientists, using automated tools, label the training examples by pointing at the features of interest, Then the system automatically learns recognizes from the examples. The diagram of the learning process is shown in Figure 8.

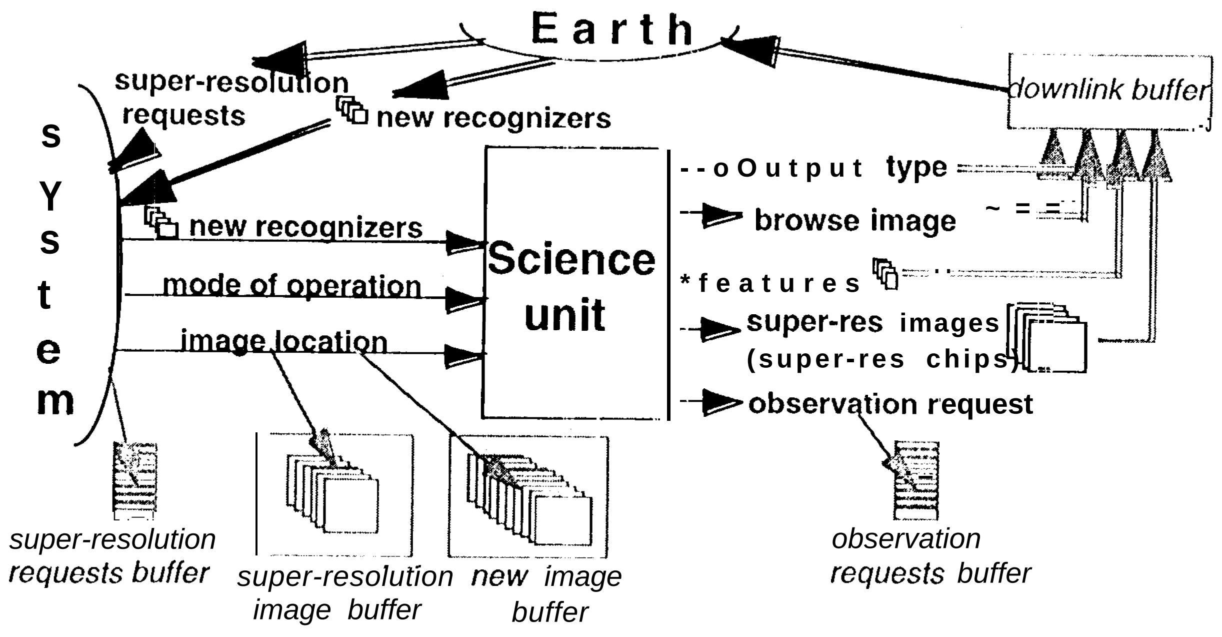

Figure 7 Science Unit Interfaces

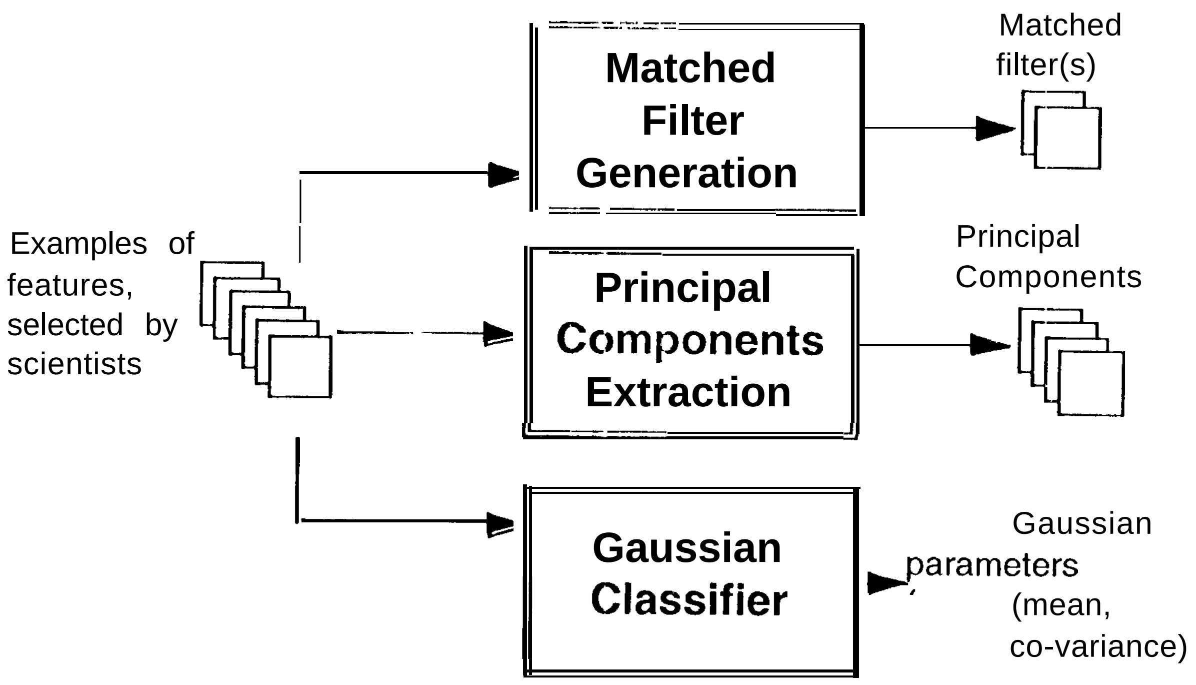
Examples of features, selected by scientists
Matched Filter Generation
Principal Components Extraction
Gaussian Classifier
Matched filter(s)
Principal Components
Gaussian parameters (mean, co-variance)

Figure 8 Off-Line Learning Process

l-he learning process takes examples labeled by scientists, processes them, and generates recognizes. There are two kinds of recognizes generated: filters(s) matched to the labeled examples, and principal components (eigenvectors) extracted from the set of training examples. Only first six most significant principal components are used. Gaussian Classifier is learned from the training examples. First, the examples are projected (correlated) onto principal components, producing feature vectors. Then the Gaussian distribution is fit onto the set of given feature vectors, producing normal distribution parameters.

The production part is performed autonomously during the S/C mission. The algorithm processes all the data in real time and finds features of interest which are buffered and downlinked to Earth. The diagram of the production process is shown in Figure 9.

The Focus Of Attention (FOA) algorithm finds possible feature locations in a short amount of time, therefore eliminating large portions of an image from detailed processing. To achieve that, FOA algorithm correlates images with the filter matched to the training examples. Principal Components Projection algorithm correlates every possible feature location with each principal component (first 6 principal components are used), producing 6-dimensional feature vectors. Some of these feature vectors correspond to true features (volcanoes), others correspond to false alarms. The Classifier separates true features from false alarms, and the coordinates of true features are returned by the process.

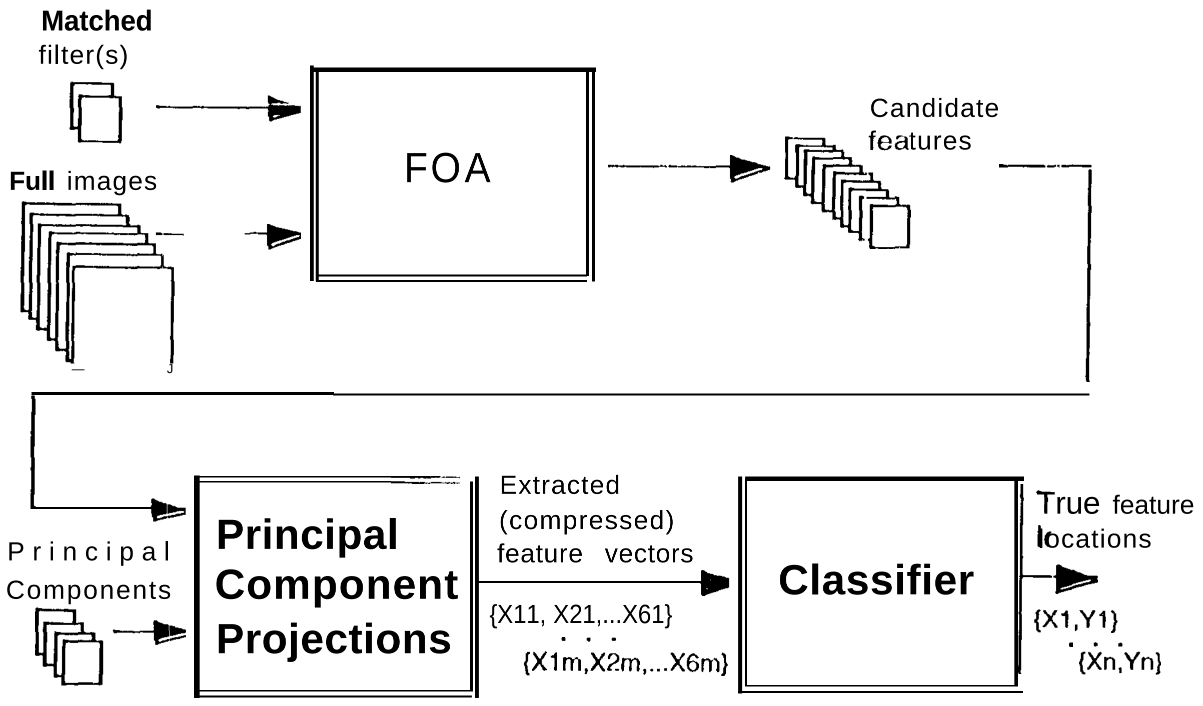

Figure 9 Real-Time Production Process

ASSAP Attitude Control and Maneuver Planning Module

The Maneuver Planner processes requests for momentum dumping or orbital change maneuvers and generates spacecraft commands for maneuver execution. A momentum dump maneuver removes excess momentum from the spacecraft reaction wheels to prevent wheel saturation. Momentum dump maneuvers are accomplished by pointing the spacecraft such that a burn of the Reaction Control Thrusters decreases the spacecraft momentum. An orbital change maneuver modifies the orbit characteristics for repeat ground tracks, nominal orbit resumption, or drag make-up. Orbital change maneuvers are accomplished by pointing the spacecraft thruster is the direction of the requested impulse, A maneuver consists of:

[1] turns to accomplish the requested pointing
[2] a momentum dump or orbital change propulsive burn at constant attitude, and
[3] turns to re-acquire a nominal spacecraft attitude.

The Maneuver Planner ensures that all spacecraft pointing constraints are satisfied during the maneuver turns and burns. These pointing constraints dictate that at no time during a maneuver shall a specified body vector/inertial vector pair point to within an a certain angular distance of each other, Figure 10.

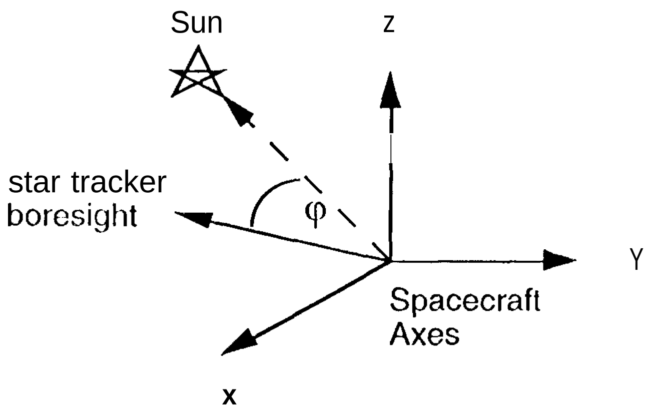

Figure 10 Typical Constraint: Sun Does Not Lie in Star Tracker Field of View (φ).

Constraints are typically designed to protect sensitive spacecraft instruments (Figure 10), to ensure spacecraft thermal limits, or to prevent excessive draining of spacecraft energy reserves.

The Maneuver Planner *a priori* computes a maneuver with the following characteristics:

[1] acquires the requested attitude at the requested epoch
[2] satisfies all active pointing constraints
[3] includes the spacecraft dynamic model[4] includes turn settling times
[5] includes turn rate limits

Maneuver Planning Task

The maneuver task is to align a body-fixed vector and an inertially fixed vector, Figure 11. This consists of rotating the spacecraft to a orientation sufficient to align the two vectors, performing a thruster(s) burn, and rotating the spacecraft to a new orientation. The basic maneuver component is a constant eigenaxis rotation with respect to an inertial frame. A turn begins from an initial stationary attitude, spin-up to a constant rotation rate about a single eigenaxis, and spin-down to a new stationary attitude, Figure 7.2. Thus all spacecraft body-fixed vectors will generate a cone trajectory about the eigenaxis, while inertial vectors will remain stationary. The magnitude of the spin vector, $\vec{\omega}$, or *turn rate* is a function of the spacecraft controller turn characteristic. A typical turn rate curve characteristic, shown in Figure 12, consists of a ramp during spin-up, a constant turn rate during coast, and a negative ramp during spin-down.

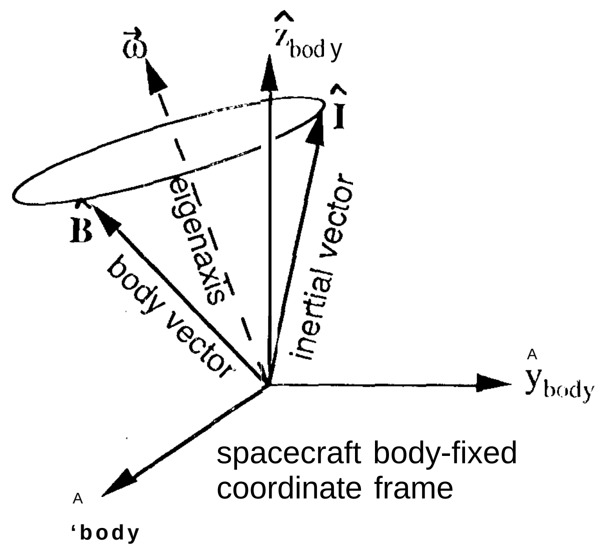

Figure 11 Eigenaxis Turn With Respect to an Inertial Frame (for simplicity the rotation of the body coordinate axes are not shown)

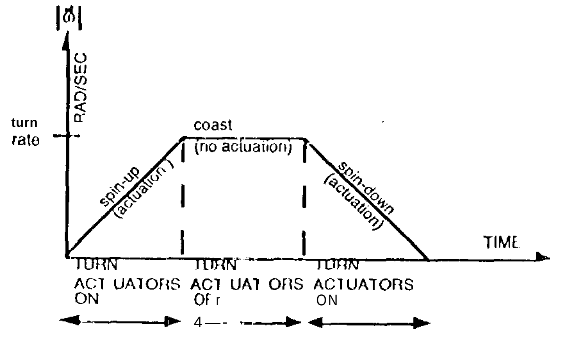

Figure 12 Typical Turn Rate Characteristic

Maneuver Planning

The planning task of the Maneuver Planner is to satisfy constraints. Most maneuvers are subject to constraints. Constraints are modeled as pointing constraints associated with a pair of vectors: a body-fixed vector and an inertially-fixed vector (Figure 10). A constraint is violated if the body-fixed vector is within a specified angular distance, or half-cone angle, of the inertially-fixed vector. Two constraints are illustrated in Figure 13.; each constraint pair is indicated by an index. There is no theoretical limit on the maximum number, n, of constraints. Each constraint associates one vector pair; i.e., a constraint violation can occur only between vectors of equal index.

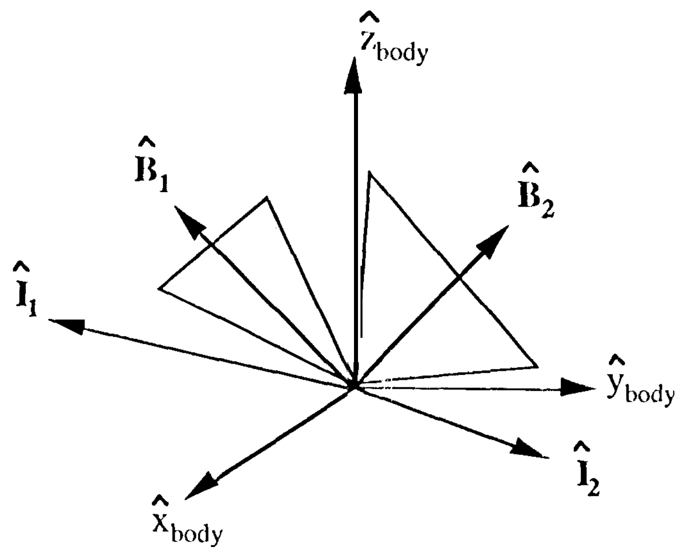

Figure 13 Two Active Constraints

The Maneuver Planner, Figure 14, consists of three main functions:

[1] initialization
[2] optimization.
[3] command generation

The Path Initializer searches for a maneuver profile that satisfies all constraints listed in the Constraint Table. The Path Initializer outputs a quaternion/timeline set (maneuver profile) representing a constraint-free maneuver. Next, the Path Optimizer adjusts the maneuver profile to locally minimize the total maneuver time. Finally, the Maneuver Commander converts the optimized maneuver profile to commands capable of driving the ASSAP turn and burn controller.

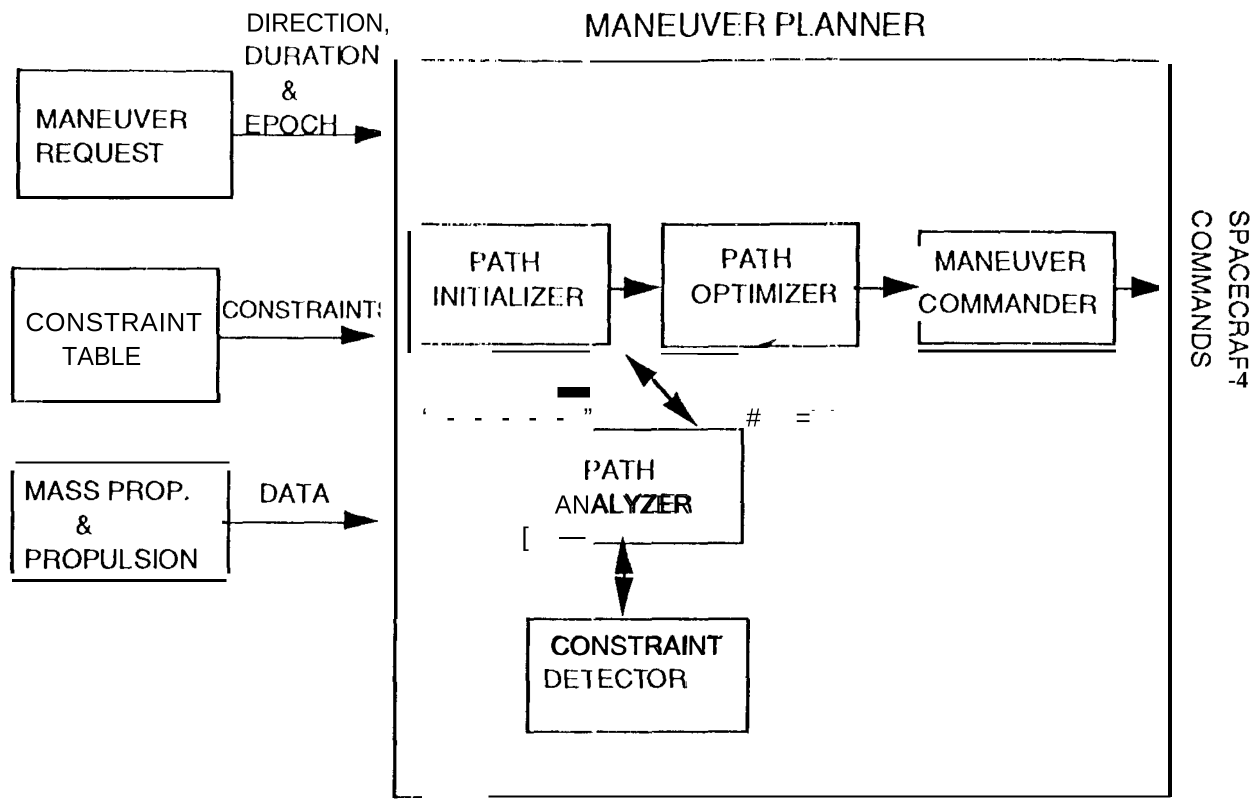

Figure 14 Maneuver Planner Block Diagram

Constraint Table

All active constraints are stored on-board. An active constraint is defined as a requirement for the next requested maneuver. Constraints are stored in a specific form denoted as the Constraint Table. The Constraint Table logically has a row for each constraint pair. Each row is in the following format:

- body unit vector in body coordinates, dimensionless (3 floating points)
- constraint unit vector in orbital or inertial frame coordinates, dimensionless (3 floating points)
- half-cone angle, radians (1 floating point)

Path Initialization

An admissible maneuver path consists of the sequence of eigenaxis turns, or maneuver profile, sufficient to align the appropriate body vector with the requested inertial vector (Figure 11). while avoiding all the constraints (Figure 13) specified in the Constraint Table. The Path Initializer first checks the shortest path that aligns the body vector, $\hat{\mathbf{B}}$, with the inertial vector, $\mathbf{I}$ If this path is constraint-free, initialization is complete.

In the general case, a constraint is violation is detected. In this case, a search for an admissible maneuver profile is initiated. Without loss of generality, each constraint is defined as a spherical circle with radius equal to the constraint half-cone angle. A spherical circle is the intersection of the unit sphere and a plane. The spherical circle is centered about the body-fixed vector, Figure 15.

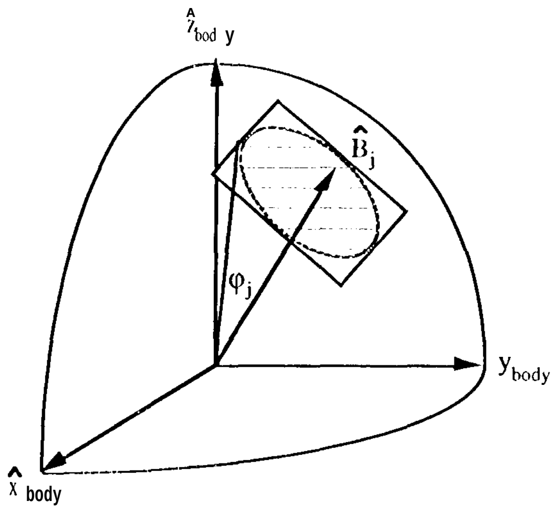

Figure 15 Constraint Modeling with Spherical Circles

The algorithm finds a path to avoid a constrain by searching on the circumference of concentric spherical circles centered about the body vector, Figure 16. The half-cone angle defines the constraint spherical circle. It is clear that any vector on the circumference is parametrized in two variables. Likewise, the family of vectors on the circumference of a concentric spherical circle is two dimensional. Define the following vectors:

$\vec{q}_1$: locates point on the constraint spherical circle where constraint is first violated; vector in the plane of the constraint spherical circle

$\vec{v}_1$: translation of $\vec{q}_1$ to concentric spherical circles; parallel to $\vec{q}_1$.

$\vec{v}_2$: completes a search basis in the plane of a concentric spherical circle.

$\hat{T}_j$ intermediate direction vector of j th constraint to avoid constraint.

$\hat{I}_j$: j th inertially-fixed unit constraint vector from Constraint Table.

$\hat{B}_j$: j th body-fixed unit constraint vector from Constraint Table.

φ_j : j th constraint half-cone angle.

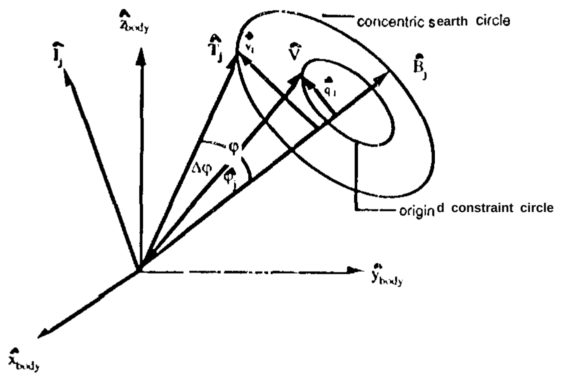

Figure 16 Constraint Avoidance with Concentric Spherical Circles

The algorithm attempts to find an intermediate attitude to avoid the constraint by aligning the inertially-fixed vector, $\hat{I}_j$, with the test vector, $\hat{T}_j$. The following equations generate the search basis with two parameters:

$$\vec{q}_1 = \hat{V} - \cos(\varphi_j)\,\hat{B}_j;$$

$$\vec{v}_1 = \left(\frac{\sin(\varphi)}{\sin(\varphi_j)}\right)\vec{q}_1;$$

$$\vec{v}_2 = \vec{v}_1 \times \hat{B}_j;$$

$$\hat{T} = \left(\cos(\eta)\,\vec{v}_1 + \sin(\eta)\,\vec{v}_2\right) + \cos(\varphi)\,\hat{B}_j$$

Next, the path from the intermediate attitude to the maneuver attitude is computed and checked for constraint violations. If constraint-free, the maneuver profile is admissible. The process if repeated for the return to nominal attitude.

Path Optimization

The Path Optimizer converses the maneuver profile to" a local minimum. This strategy allows the initialization to proceed without regard to optimality. Let q denote a quaternion and T denote spacecraft time. Then a typical maneuver profile 'is illustrated in Figure 17. The function of the intermediate attitudes, "A" and "B", is to avoid the constraints. These quaternions are computed in the Path Initializer.

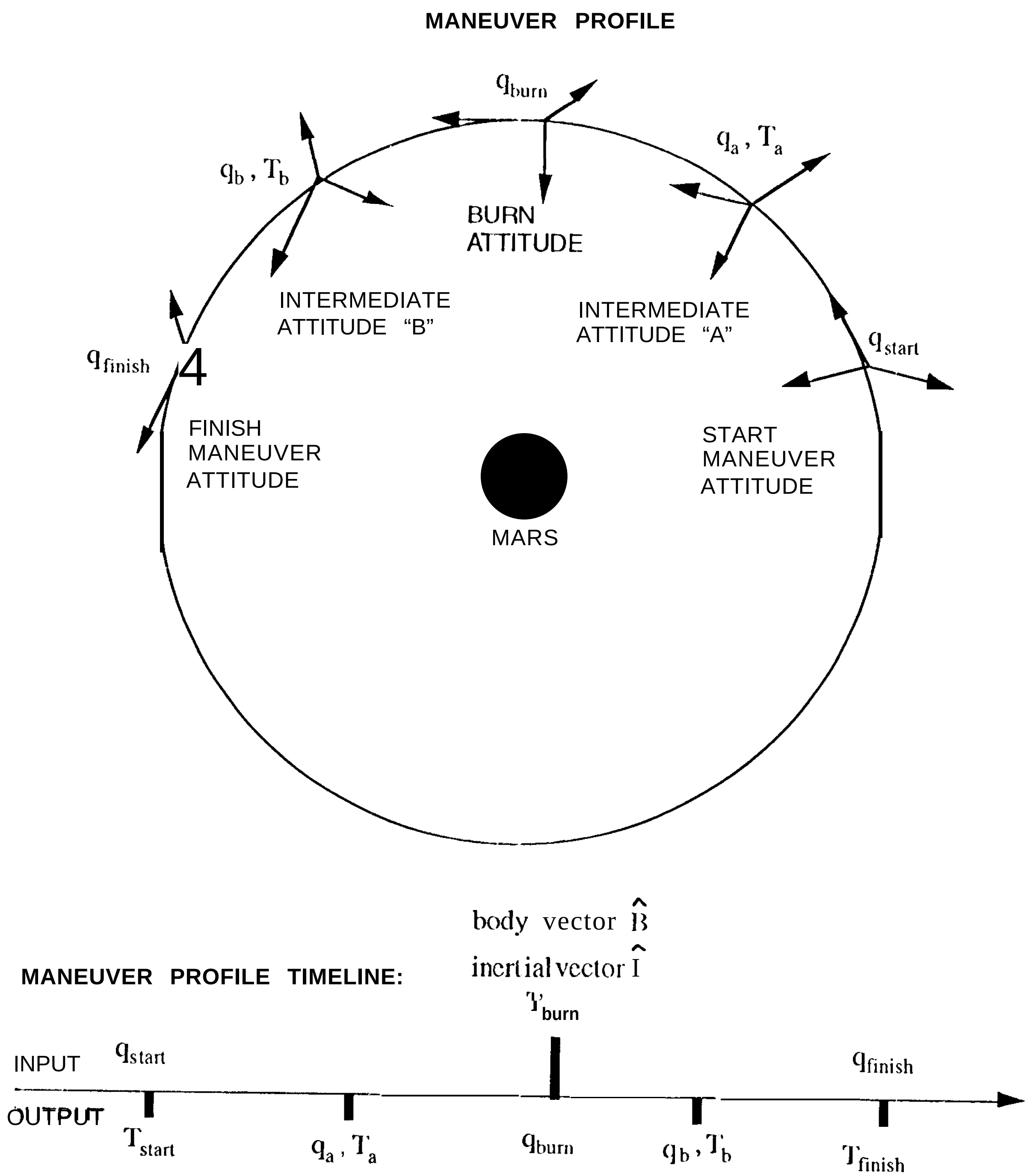

Figure 17 The Maneuver Profile is Defined By Attitude Quaternions

Local minima are defined with respect to a specific cost function. The ASSAP cost function is computed as follows:

$$j = (T_{finish} - T_{start}) + \lambda T_a$$

where

$$T_a = \sum_{j=1}^{n} (\text{time inside constraint cone for j th constraint})$$

The term T_a is a penalty function which causes the cost function j to peak if the maneuver is adjusted into a constraint and λ is a sensitivity multiplier. Thus the Path Optimizer functions to cause the trajectories to be a direct as possible until a constraint spherical circle is tangentially intersected by a body vector.

Six degrees of freedom exist to plan a maneuver: the two intermediate attitudes q_a and q_b. Let the independent coordinates of the quaternions be indicated as:

$$q_a = [x_1, x_2, x_3]^t;$$

$$q_b = [x_4, x_5, x_6]^t;$$

$$\vec{x} = [x_1, x_2, X3,X4,X5, x_6]^t;$$

where $\vec{X}$ is the maneuver profile. Then the cost function and its gradient with respect to the maneuver profile is:

$$j = j(\vec{x});$$

$$\nabla j = \left[\frac{\partial j}{\partial x_1}, \frac{\partial j}{\partial x_2}, \frac{\partial j}{\partial x_3}, \frac{\partial j}{\partial x_4}, \frac{\partial j}{\partial x_5}, \frac{\partial j}{\partial x_6}\right]^t$$

The gradient vector points in the direction of steepest descent to minimize the cost function ($\bar{j}$). The Path Optimizer converges the maneuver profile ($\vec{X}$) via the following dynamics:

$$j = j(\vec{x});$$

$$\nabla j = \left[\frac{\partial j}{\partial x_1}, \frac{\partial j}{\partial x_2}, \frac{\partial j}{\partial x_3}, \frac{\partial j}{\partial x_4}, \frac{\partial j}{\partial x_5}, \frac{\partial j}{\partial x_6}\right]^t$$

In Figure 18., the converge law integrator terminates at a local minima of the cost function.

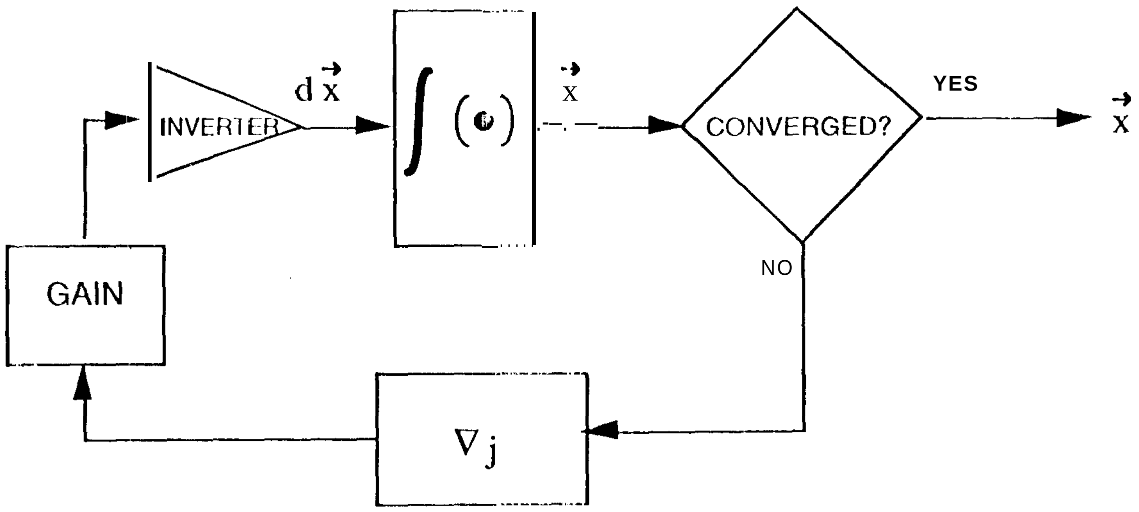

Figure 18 Path Optimizer Maneuver Profile Convergence Law

-Maneuver Commander

The Maneuver Commander is the interface to the spacecraft controller. This module is normally specific to the particular spacecraft. In general, the maneuver profile is utilized to compute a series of commanded quaternions and body coordinate rate vectors sequenced over time. The Maneuver Planner produces eigenaxis turns and therefore are represented as an axis and a turn angle, with the turn angle sequenced over time. The turn angle is time-varying and is the integral of the turn rate characteristic (Figure 12). Thus a turn between two inertial attitudes is sequenced over time as:

$$q(t) = q^{a}_{start}(t)\, q^{start}_{inertial};$$

$$q^{a}_{start}(t) \quad \begin{vmatrix} \sin\left(\frac{\theta(t)}{2}\right) e_x \\ \sin\left(\frac{\theta(t)}{2}\right) e_y \\ \sin\left(\frac{\theta(t)}{2}\right) e_z \\ \cos\left(\frac{\theta(t)}{2}\right) \end{vmatrix}$$

$$\theta(t) = \text{integral of turn rate characteristic}$$

$$\begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} = \text{eigenaxis of turn}$$

$$q^{start}_{inertial} = \text{inertial attitude at start of turn}$$

The above process is utilized for each turn, with applicable settling and burn times inserted.

C.on.elusions

The ASSAP task has been underway for approximately one year at the time of this writing and has made three major milestones.

Integrated Prototype

An integrated prototype system that demonstrates onboard autonomy has been developed. It consists of the following features:

- 1 ask Control Architecture (TCA),
- Flight rules and constraints,
- High level command dictionary,
- Dynamic spacecraft/world truth model,
- Attitude determination and control system (ADCS),
- inertial vector propagation command structure,
- Provision for fault injection,
- On- screen spacecraft animation ,
- Graphical user interface, and
- Modular design for portability to Unix/FST/VxWorks/multi-processor systems.

The prototype consists of four main components the planner and TCA executive, the domain specific modules, the fault analyzer and the spacecraft dynamics model. The planner and TCA executive is at the heart of the autonomy capability. It is here that the high-level commands are converted into an executable sequence which is in compliance with the flight rules and constraints.

The domain specific modules are specialized modules providing a specific capability such as orbit determination, trajectory planning, propulsive maneuver and science feature recognition. A fault analyzer keeps track of the system configuration, mode, health and the status of requested commands. When an anomaly is detected it figures out the corrective action needed and submits appropriate commands to the executive.

Flight _.Demonstration of Autonomous Maneuver Module

'i his is the most mature of the software modules under development. I-he feasibility for autonomous propulsive maneuver sequence generation was demonstrated a year ago. This year 's effort generalized the algorithms allowing arbitrary Euler turns in place of the single axis turns and the code is being written to comply with flight software code requirements. Section 7 describes the propulsive maneuver module.

1 he DV module is partitioned to include pre-maneuver and post-maneuver commands. The pre-maneuver commands ensure that the spacecraft is in the correct software mode and that hardware preparations such as calibration and warm-up are accomplished prior to the maneuver "execution. The post-maneuver

commands are needed to return the spacecraft state back to nominal after the maneuver execution.

1 his module under separate funding is being readied for possible integration with the Cassini flight software. Also under separate funding this module is being customized to be demonstrated in flight as an experiment on the TOPEX spacecraft which is currently in the extended mission phase.

Onboard Navigation Module Allows Science Observation Redirection

This work will demonstrate an integration of the science observation scheduling process with the navigation system. The scope of the current delivery is restricted and concentrates on computing and delivering requested AV'S, spacecraft position and velocity information, and science observation times.

The navigation module receive a list of features of interest for super-resolution imaging. The location of these features is assigned to one or more of 30 possible ground track bins. Based upon the current orbit knowledge, the times for entry into each of the bins is obtained. This knowledge is passed back to the science module which chooses the next the entry time and hence the bin it wants the orbiter to enter. The chosen bin identification is passed back to the navigation module which then schedules maneuvers to enter and exit the 13 orbit repeat ground track bin. In addition, the landmark flyover times are re-computed using the latest trajectory information to support scheduling of the SAR observations.

Acknowledgments

The research described in this paper was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.