Automatic Commanding of the Mars Observer Camera

Michael Caplinger
Malin Space Science Systems, Inc.
PO Box 910148
San Diego CA 92191-0148, USA
phone: (619) 552-2650 fax: (619) 458-0503
mc@msss.com

KEY WORDS AND PHRASES

Conflict resolution, spacecraft imaging instrument operations, mission planning, sequencing and scheduling.

INTRODUCTION

Mars Observer, launched in September 1992, was intended to be a "survey-type" mission that acquired global coverage of Mars from a low, circular, near-polar orbit during an entire martian year [1]. As such, most of its instruments had fixed data rates, wide fields of view, and relatively low resolution, with fairly limited requirements for commanding. An exception is the Mars Observer Camera, or MOC. The MOC consists of a two-color Wide Angle (WA) system that can acquire both global images at low resolution (7.5 km/pixel) and regional images at commandable resolutions up to 250 m/pixel. Complementing the WA is the Narrow Angle (NA) system, that can acquire images at 8 resolutions from 12 m/pixel to 1.5 m/pixel, with a maximum crosstrack dimension of 3 km. The MOC also provides various forms of data compression (both lossless and lossy), and is designed to work at data rates from 700 bits per second (bps) to over 80k bps [2].

Because of this flexibility, developing MOC command sequences is much more difficult than the routine mode-changing that characterizes other instrument operations. Although the MOC cannot be pointed (the spacecraft is fixed nadir-pointing and has no scan platform), the timing, downlink stream allocation, compression type and parameters, and image dimensions of each image must be commanded from the ground, subject to the constraints inherent in the MOC and the spacecraft. To minimize the need for a large operations staff, the entire command generation process has been automated within the MOC Ground Data System [3].

Following the loss of the Mars Observer spacecraft in August 1993, NASA intends to launch a new spacecraft, Mars Global Surveyor (MGS), in late 1996. This spacecraft will carry the MOC flight spare (MOC 2). The MOC 2 operations plan will be largely identical to that developed for MOC, and all of the algorithms described here are applicable to it.

TARGET GENERATION

In advance, users define "time-independent observing plans" that consist of a specification of an area or feature edge on the surface of Mars, the type of acquisition(s) to be made there, any geometric and timing constraints (such as lighting angles, season, etc.), the image size, resolution, allowable compression types, and a single number indicating the priority of the observation. (By convention, priorities are non-negative and the more important an observation is, the larger its priority number.)

Daily during operations, these plans and spacecraft position are examined and a list of potential images is generated. This "strawman sequence" consists of image acquisition commands to be sent to the MOC; each command specifies the time a specific optical system is to be activated, and a set of parameters (image size, resolution, compression type, and downlink channel assignment) to be associated with that particular acquisition.

Since there are typically many thousands of active observing plans, and targeting is performed frequently, the algorithm that generates the strawman sequence must be very efficient. The algorithm we finally used treats NA and WA swaths in two different ways. Such a dual approach makes sense, since the clocking rates, and hence the accuracy requirements, are two orders of magnitude different between NA and

WA. Also, the very narrow field of view of the NA allows a much simpler geometric description of its swath to be used.

Narrow Angle

The NA swath is treated as a widthless polyline generated by sampling the ground track in equal time intervals and assuming linearity in lat/lon space between these points. The required accuracy is obtainable with fixed 5-second spacing, though a method using variable spacing, with more time resolution near the poles, is preferable and would be required for non-circular orbits.

The core of the algorithm is a loop over each segment of the ground track. A clip test between this line and the target box is done in lat/lon space, and the locations of the intersection(s), if any, are calculated, using the parametric representation of the line segment. These parameters are used to compute the first and last times of intersection, and the loop gathers the minimum and maximum time values. These values are used to generate the start time and dimensions of the image event, if there were any intersections. If there are no intersections, this area is not accessible on this orbit.

Because the majority of boxes on a given run are probably not accessible by the NA, some attention was paid to rejecting a line segment that did not intersect the box as soon as possible in the algorithm, using a trivial bounding-box calculation.

Wide Angle

The WA swath covers over 30 degrees of longitude and obviously cannot be treated as a widthless line. Instead, two ground track polylines are calculated -- one representing the maximum view angle of the WA in the +Y direction (plus) and the other in the -Y direction (minus). These curves represent the overall field of view of the MOC. Note that during the ascending part of the orbit, plus is east of minus; during the descending part, minus is east of plus. This ordering is used to remove the meridian-crossing ambiguity.

Rather than process the entire swath as a single polygon, it is broken up into four-sided quadrilaterals, called "quads", with sides connecting the four points defined by the plus and minus tracks at time t and t+delta. (Note that the lines connecting the plus and minus tracks at the same time are not the tracks followed by single scanlines, except near the equator, and they are not used as approximations to scanlines -- they are merely arbitrary lines.) Because quads are always convex, processing them is relatively simple.

The basic algorithm is a loop through all of the quads for a given orbit. At each point, the quad is tested against a target box. Two kinds of tests are performed and point coordinates are recorded. First, any box corner contained within the quad is recorded. Second, any point of intersection between the plus and minus edges of the quad with the box are recorded.

After all the quads are compared against a given box, the gathered test points are mapped to times and WA pixel coordinates using a separate iterative algorithm. (If a box generated no test points with the swath, it cannot be viewed on that orbit.) The ranges of the test point pixels and times are recorded and used to generate the timing and dimensions for a WA event.

The algorithm described so far ignores latitudes near the pole, because quads do not appear above a given critical latitude. (For example, suppose a quad had its bottom edge at latitude 85, and then the spacecraft passed over the pole and back down to latitude 85 before the quad's top edge was created. Then the quad would appear to have no latitude extent above latitude 85.) The polar areas can be handled by noting that if an orbit changes from ascending to descending or vice versa near a pole, then all boxes above a certain latitude are seen on that pass. In addition, if an orbit passes sufficiently close to a pole, then all boxes above a given latitude in a range of 180 degrees of longitude are also seen. Each pole can fall into at most one of these two categories (it is physically impossible to go from ascending to descending and back on the same orbit because the placement of the terminator cannot change that rapidly.) Thus, every vertex of a box that occurs in one of these polar regions is added to the list of test points.

Performance

Our initial performance goal was to process 3000 potential target areas for one orbit in less than 5 minutes. Operationally, we saw an average time of 2.1 minutes for this task on the Sun SPARCstation IPX (a roughly 20

SPECmark system), so we have exceeded our goal by over a factor of two. However, since we were processing a significantly larger task (13 orbits and nearly 10,000 plans) total time was about 1.5 hours, which can become burdensome.

Many solutions are possible without changing the algorithm, the simplest of which is to use a faster processor. Also, the algorithm is easily done in parallel either by splitting plans or orbits across processors, and so would benefit from the multiprocessor systems becoming available. We predict that a four-processor SPARCstation 10 system could perform the task above in less than 15 minutes.

CONFLICT RESOLUTION

Unfortunately, not all of the commands in a strawman sequence can be executed because of limited instrument resources. These resources include buffer space, CPU processing time, downlink rate, and power. Since the timing for each acquisition is fixed by the spacecraft's position, the sequence cannot be reordered. (This makes the MOC sequencing problem fundamentally different from other space application sequencing problems, such as Voyager-like or Hubble Space Telescope sequencing [4,5].) The only free parameters left to modify are whether or not to acquire each potential image, and the compression type and downlink channel assignment for each image. (While it is possible for science users to restrict compression or downlink channel to particular values, this may place limits on how well the automatic process can optimize the overall sequence. In some cases, resolution or image size can be altered as well, but the automatic program does not attempt such modifications.)

Thus, the MOC sequencing program seeks to maximize the number of images taken from the input sequence, while choosing images of higher priority, all other things being equal.

Obviously, the key to solving the problem is to generate alternative possible sequences and see which have conflicts. A critical problem is how to know if a given MOC sequence fits within the resource constraints. The solution is a fast event-driven simulator that mimics the behavior of the instrument and detects resource conflicts. Using this simulator as a black box, it can be determined if a given sequence is conflict-free, and if not, when and what the conflict is.

Additionally, we have the following desires for the algorithm:

(1) it should be applicable across all data rates (data rate assignments have changed several times and can be expected to change again)
(2) it should be insensitive to exact details of instrument behavior (during development, the performance and details of instrument operation were not known to any accuracy)
(3) it should allow "splicing" of daily sequences because planning is done piecemeal, not all at once

Our initial approach was to develop a series of heuristics that took a full input sequence and deleted or modified individual items until the sequence was without conflict. Though this worked after a fashion, it was extremely slow, because there was no systematic way to search for alternatives. Therefore, it was decided to invert the approach and develop a series of heuristics to take an initially empty input sequence and add items to it until no more can be added.

By "heuristic", we mean a rule intended to choose a favorable outcome without any analytical evidence that such an outcome would be chosen. One could have very specific heuristics, such as "when the data rate is higher than X, use predictive compression", or quite general heuristics, such as "choose the alternative such that the image is resident in the buffer for the shortest period of time." The more general heuristics are preferable, since they rely on less knowledge of the specifics of the process. In addition, specific heuristics may be derivable from the general heuristics, such that a system using only the general heuristic will appear to be operating under the specific heuristics as well.

In fact, we have obtained good results with a single heuristic, which we call "shortest-residence-time". This is used by the following algorithm:

```
sequence = ε (empty sequence)
for priority = highest to lowest
 for images at this priority ordered by time,
 earliest to latest
  for each alternative
    given sequence so far, compute residence time
    of current image in instrument for this
    alternative. If alternative generates conflict,
    set time to ∞
```

if any time is not ∞, add this image, using the shortest-residence-time alternative, to sequence

The residence time of an image is the amount of time any fraction of either raw image data or any compressed or processed version of that data is stored in the MOC buffer. The alternatives examined by the program currently are from the set {predictive compression, channel 1; transform compression, channel 1; predictive compression, channel 2; transform compression, channel 2}; obviously, other alternatives could be easily added.

Performance

The requirement set for the MOC GDS was that conflict resolution for a 12-orbit strawman sequence containing 1500 potential acquisitions could be performed in less than 5 minutes; the existing system meets this performance goal on a Sun SPARCstation 1 (a roughly 10 SPECmark system.)

To give an idea of the size of a typical problem and the effectiveness of our algorithm, our standard test target set contains about 2500 planned areas. For a twelve-orbit period chosen at random, 111 images (50 WA and 61 NA images) were found to be accessible. At low data rate, 29 of the 111 images could be taken; at high data rate with 4 orbits of realtime passes, 75 of the images could be taken.

The relationship between the sequences found by our software and optimal sequences is not known, although the general problem has been shown to be NP-complete, meaning that the optimal sequence cannot be found without examining *all* sequences. We do note that our sequences utilize 90% or more of the available resources, indicating that little waste is present. For small sequences (about length 10) for which the optimal sequence could be found, our algorithm either finds the optimal sequence or at worst, fails to take one image.

CONCLUSIONS

The existing system is operational and has processed hundreds of simulated sequences that were then executed on the actual hardware (in ground testing) without conflict. We hope to use this system for instrument operations when Mars Global Surveyor goes into orbit around Mars in late 1997.

Some simple additions would make it possible to extend this system to missions in eccentric orbits, such as the "transition orbit" of MGS. These additions include the provision of a resolution requirement for time-independent plans, and removal of the reliance on geometric properties of the ground track for simplification of the targeting algorithm. While these additions would not completely solve the problem for missions which use a scan platform or spacecraft slewing to point their instruments, we believe this framework would be easily applicable even to such missions, by using a series of heuristics and resolution requirements to fix observations in time. We expect to experiment with this approach soon.

ACKNOWLEDGEMENTS

I especially thank Mike Malin, the MOC Principal Investigator, without whom the MOC would have never existed. Marc Sarrel implemented the first version of the targeting algorithm; Jeff Warren provided valuable insight into aspects of MOC geometry. This work was supported by the Jet Propulsion Laboratory, Contract 959060 to Malin Space Science Systems, for development and operation of the Mars Observer Camera.

REFERENCES

[1] Albee, A.L., *et al*, 1992. Mars Observer Mission. *Journal of Geophysical Research* 97(E5):7665-7680.

[2] Malin, M.C., *et al*, 1991. Design and Development of the Mars Observer Camera. *International Journal of Imaging Systems and Technology* 3(1):76-91.

[3] Caplinger, M., 1993. The Mars Observer Camera Ground Data System. In *Proceedings of the Ninth AIAA Conference on Computers in Aerospace*, San Diego, CA.

[4] Dias, W.C., *et al*, 1987. PLAN-IT: Scheduling Assistant for Solar System Exploration. *Telematics and Informatics* 4(4):275-287.

[5] Miller, G., *et al*, 1987. Expert Systems Tools for Hubble Space Telescope Observation Scheduling. *Telematics and Informatics* 4(4):301-311.