A navigation software system that implements these algorithms generates a simple graphical user interface, through which the user can specify either waypoints in two or three dimensions or the ground area to be surveyed. Alternatively, the user can load a data file containing waypoint coordinates. The user can also specify other parameters that affect the planned trajectory, including the field of view of the camera and the dynamical parameters, the primary one being the minimum allowable turn radius of the aerobot. Then assuming constant airspeed, the algorithms compute a minimum-time or minimum-length trajectory that takes account of all of the aforementioned requirements and constraints. Notably, in one of the algorithms, the turning dynamics of the aerobot are represented by a cubic spline that is used to interpolate the trajectory between waypoints.

In some contemplated future versions, the need for intervention by human users would be reduced: Waypoints specified by users could be supplanted by data generated by onboard artificial-intelligence image-data-processing systems programmed to strive to satisfy mission specifications.

This work was done by Eric Kulczycki and Alberto Elfes of Caltech and Shivanjli Sharma of University of California at Davis for NASA's Jet Propulsion Laboratory.

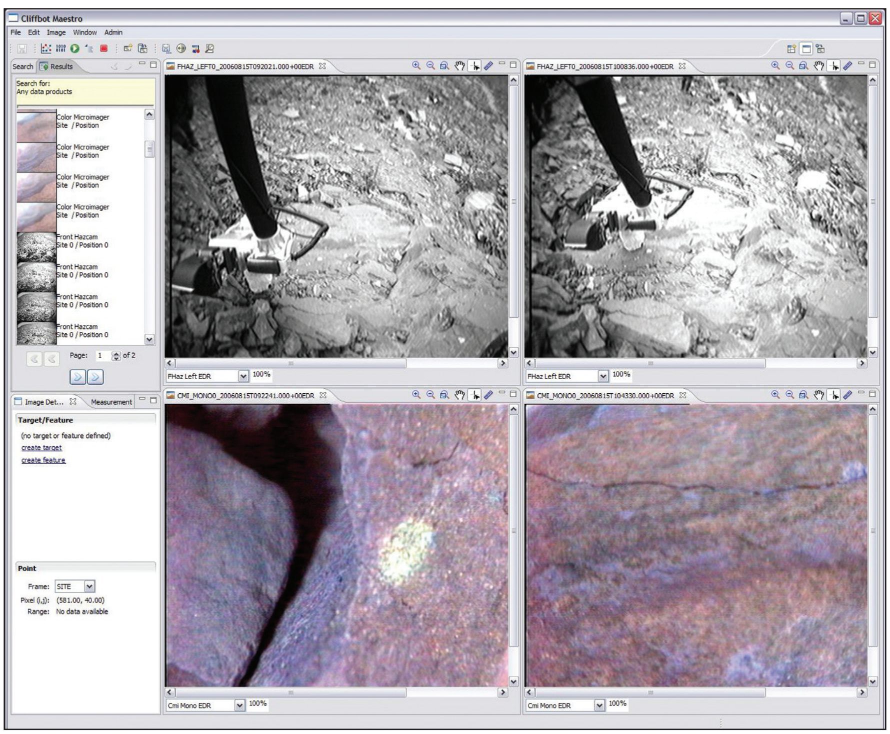
The software used in this innovation is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-44395.

Cliffbot Maestro

NASA's Jet Propulsion Laboratory, Pasadena, California

Cliffbot Maestro (see figure) permits teleoperation of remote rovers for field testing in extreme environments. The application user interface provides two sets of tools for operations: stereo image browsing and command generation.

The stereo image-browsing feature allows the operator to see images in either 2D or 3D views. This is useful in order to develop a route for the rover to safely drive, as well as identify interesting objects for scientific exploration.



Cliffbot Maestro User Interfaces.

The command-generation tool is used to author a script (using either a drag & drop interface, or a textual command-line one) and send it to the rover. These scripts are not only for driving, but also for sample collection, reconnaissance, imaging, and science data acquisition.

The software runs on dedicated hardware that can withstand extremely cold temperatures. Its test bed is a Panasonic Toughbook (or equivalent) rugged laptop operating in the deep Arctic for extended periods. While the hardware doesn't have to be cutting-edge, it must withstand continued cold.

Cliffbot Maestro also provides engineering metrics about the state of the rover, in order to monitor its health, as well as the condition of the robotic arm. This allows for remote support while in the field.

This work was done by Jeffrey S. Norris, Mark W. Powell, Jason M. Fox, Thomas M. Crockett, and Joseph C. Joswig of Caltech for NASA's Jet Propulsion Laboratory.

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-46433.

Tracking Debris Shed by a Space-Shuttle Launch Vehicle

Lyndon B. Johnson Space Center, Houston, Texas

The DEBRIS software predicts the trajectories of debris particles shed by a space-shuttle launch vehicle during ascent, to aid in assessing potential harm to the space-shuttle orbiter and crew. The user specifies the location of release and other initial conditions for a debris particle. DEBRIS tracks the particle within an overset grid system by means of a computational fluid dynamics (CFD) simulation of the local flow field and a ballistic simulation that takes account of the mass of the particle and its aerodynamic properties in the flow field. The computed particle trajectory is stored in a file to be postprocessed by other software for viewing and analyzing the trajectory.

DEBRIS supplants a prior debris-tracking code that took ≈15 minutes to calculate a single particle trajectory: DEBRIS can calculate 1,000 trajectories in ≈20 seconds on a desktop computer. Other improvements over the prior code include adaptive time-stepping to ensure accuracy, forcing at least one step per grid cell to ensure resolution of all CFD-resolved flow features, ability to simulate rebound of debris from surfaces, extensive error checking, a built-in suite of test cases, and dynamic allocation of memory.

This program was written by Phillip C. Stuart of Johnson Space Center and Stuart E. Rogers of Ames Research Center. Further information is contained in a TSP (see page 1). MSC-23945-1

Estimating Thruster Impulses From IMU and Doppler Data

NASA's Jet Propulsion Laboratory, Pasadena, California

A computer program implements a thrust impulse measurement (TIM) filter, which processes data on changes in velocity and attitude of a spacecraft to estimate the small impulsive forces and torques exerted by the thrusters of the spacecraft reaction control system (RCS). The velocity-change data are obtained from line-of-sight-velocity data from Doppler measurements made from the Earth. The attitude-change data are the telemetered from an inertial measurement unit (IMU) aboard the spacecraft.

The TIM filter estimates the three-axis thrust vector for each RCS thruster, thereby enabling reduction of cumulative navigation error attributable to inaccurate prediction of thrust vectors. The filter has been augmented with a simple mathematical model to compensate for large temperature fluctuations in the spacecraft thruster catalyst bed in order to estimate thrust more accurately at deadbanding "cold-firing" levels. Also, rigorous consider-covariance estimation is applied in the TIM to account for the expected uncertainty in the moment of inertia and the location of the center of gravity of the spacecraft. The TIM filter was built with, and depends upon, a sigma-point consider-filter algorithm implemented in a Python-language computer program.

This program was written by Michael E. Lisano and Gerhard L. Kruizinga of Caltech for NASA's Jet Propulsion Laboratory.

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-45825.

Oxygen Generation System Laptop Bus Controller Flight Software

Lyndon B. Johnson Space Center, Houston, Texas

The Oxygen Generation System Laptop Bus Controller Flight Software was developed to allow the International Space Station (ISS) program to activate specific components of the Oxygen Generation System (OGS) to perform a checkout of key hardware operation in a microgravity environment, as well as to perform preventative maintenance operations of system valves during a long period of what would otherwise be hardware dormancy. The software provides direct connectivity to the OGS Firmware Controller with pre-programmed tasks operated by on-orbit astronauts to exercise OGS valves and motors. The software is used to manipulate the pump, separator, and valves to alleviate the concerns of hardware problems due to long-term inactivity and to allow for operational verification of microgravity-sensitive components early enough so that, if problems are found, they can be addressed before the hardware is required for operation on-orbit.

The decision was made to use existing on-orbit IBM ThinkPad A31p laptops and MIL-STD-1553B interface cards as the hardware configuration. The software at the time of this reporting was developed and tested for use under the Windows 2000 Professional operating system to ensure compatibility with the existing on-orbit computer systems.

This program was written by Chad Rowe and Donna Panter for Johnson Space Center. Further information is contained in a TSP (see page 1). MSC-24316-1

Port-O-Sim Object Simulation Application

Goddard Space Flight Center, Greenbelt, Maryland

Port-O-Sim is a software application that supports engineering modeling and simulation of launch-range systems and subsystems, as well as the vehicles that operate on them. It is flexible, distributed, object-oriented, and real-time. A scripting language is used to configure an array of simulation objects and link them together. The script is contained in a text file, but executed and controlled using a graphical user interface.

A set of modules is defined, each with input variables, output variables, and settings. These engineering models can be either linked to each other or run as standalone. The settings can be modified during execution.