Software

Autonomous Instrument Placement for Mars Exploration Rovers

Autonomous Instrument Placement (AutoPlace) is onboard software that enables a Mars Exploration Rover to act autonomously in using its manipulator to place scientific instruments on or near designated rock and soil targets. Prior to the development of AutoPlace, it was necessary for human operators on Earth to plan every motion of the manipulator arm in a time-consuming process that included downlinking of images from the rover, analysis of images and creation of commands, and uplinking of commands to the rover. AutoPlace incorporates image analysis and planning algorithms into the onboard rover software, eliminating the need for the downlink/uplink command cycle. Many of these algorithms are derived from the existing ground-based image analysis and planning algorithms, with modifications and augmentations for onboard use.

AutoPlace also utilizes pre-existing onboard arm control, arm collision-detection, and stereoscopic image processing software. In addition, to satisfy needs specific to the Mars Exploration Rovers and to increase safety, AutoPlace incorporates a volumetric terrain visibility analysis algorithm, a uniform target selection algorithm, and a template-based trajectory generation algorithm that were not parts of the prior onboard or ground software.

This program was written by P. Chris Leger and Mark Maimone of Caltech for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-44820.

Mission and Assets Database

Mission and Assets Database (MADB) Version 1.0 is an SQL database system with a Web user interface to centralize information. The database stores flight project support resource requirements, view periods, antenna information, schedule, and forecast results for use in mid-range and long-term planning of Deep Space Network (DSN) assets.

Project requirements can be entered using interval-based patterns, which allow planning analysts to capture project requirements more accurately. Project information can be stored in such a way as to allow multiple sets to be entered for various scenario studies. For example, a mission can have multiple view periods and many sets of requirements. This extends to schedules and forecasts as well. The Web component of this system allows users to modify this information and to generate graphical and tabular reports from it.

Unlike other toolsets used previously, MADB allows the user to enter requirements in the most flexible way. It also allows for many view periods for each project as well as a hierarchy system for classifying them. MADB-generated reports can span multiple projects and view periods. MADB uses an industry-standard SQL database, which enables future generic software improvement and multiple levels of access. The Web interface also can be accessed from any platform.

The RAPS TIGRAS and DRAGON tools are tightly integrated with MADB. Together they are used by the Resource Allocation Planning Service to schedule and forecast DSN assets.

This program was written by John Baldwin, Silvino Zendejas, Sandy Gutheinz, Chester Borden, and Yeou-Fang Wang of Caltech for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-44714.

TCP/IP Interface for the Satellite Orbit Analysis Program (SOAP)

The Transmission Control Protocol/Internet protocol (TCP/IP) interface for the Satellite Orbit Analysis Program (SOAP) provides the means for the software to establish real-time interfaces with other software. Such interfaces can operate between two programs, either on the same computer or on different computers joined by a network. The SOAP TCP/IP module employs a client/server interface where SOAP is the server and other applications can be clients. Real-time interfaces between software offer a number of advantages over embedding all of the common functionality within a single program. One advantage is that they allow each program to divide the computation labor between processors or computers running the separate applications. Secondly, each program can be allowed to provide its own expertise domain with other programs able to use this expertise.

For example, a telemetry acquisition system can handle the complexity of downloading data from the satellite, whereas SOAP can use such data to offer 3D displays and status information in a human-readable form. Both programs can operate efficiently, especially when they are hosted on separate machines. The SOAP TCP/IP interface supports the same rich command structure that its input file parser provides. The input is obtained and processed through the network instead of through files.

In addition, SOAP can bring additional analytical resources to bear against incoming data streams. SOAP can process these TCP/IP streams in a number of ways. It can access two simultaneous streams from different sources. This provides an added degree of flexibility in a real-time environment. The software can remain interactive while receiving data, allowing the user to configure different 3D views and data displays. For instance, a user can examine a 3D model based on orientation data streaming in from an independent source, such as the telemetry feed, or a robotic simulation such as the FORESIGHT software. Additionally, the SOAP TCP/IP interface can be configured in batch mode and reside on a server. Because SOAP can be tasked to automatically generate image and data files, it can be used to set up an automated Web site offering near real-time image and data reporting.

This software is portable and runs on four separate computer platforms: MS-Windows, Mac OS X, Linux, and Sun Solaris. There are no minimum hardware requirements other than that to run the host operating system. The host system should be capable of running OpenGL applications. SOAP performs best on the machines with good graphics hardware acceleration.

This work was done by Robert Carnright of Caltech and David Stodden and John Coggi of The Aerospace Corporation 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-45056.

Trajectory Calculator for Finite-Radius Cutter on a Lathe

A computer program calculates the two-dimensional trajectory (radial vs. axial position) of a finite-radius-of-curvature cutting tool on a lathe so as to cut a workpiece to a piecewise-continuous, analytically defined surface of revolution. (In the original intended application, the tool is a diamond cutter, and the workpiece is made of a crystalline material and is to be formed into an optical resonator disk.) The program also calculates an optimum cutting speed as F/L, where F is a material-dependent empirical factor and L is the effective instantaneous length of the cutting edge.

The input to the program includes the analytical specification of each desired continuous piece of the surface. The output of the program corresponds to an approximate tool trajectory in the form of (1) a set of short straight-line segments connecting the precise trajectory points at user-defined axial steps and (2) the optimum cutting speed for each segment. The program includes algorithms for rounding corners, limiting the depth of cut, and making extra cutouts to prevent excessive stresses. The output of this program is read by a different program that controls stepping motors that move the cutting tool.

This program was written by Dmitry Strekalov, Anatoliy Savchenkov, and Nan Yu of Caltech for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-45086.

Integrated System Health Management Development Toolkit

This software toolkit is designed to model complex systems for the implementation of embedded Integrated System Health Management (ISHM) capability, which focuses on determining the condition (health) of every element in a complex system (detect anomalies, diagnose causes, and predict future anomalies), and to provide data, information, and knowledge (DIaK) to control systems for safe and effective operation.

An important functionality of ISHM is that DIaK is embedded and easily accessible. The software includes tools for distributed storage, evolution, and distribution of DIaK, and easy accessibility. For example, an intelligent sensor includes a TEDS (Transducer Electronic Data Sheet); processes for data validation and sensor health determination; communication capability to provide DIaK to other elements of the system; and to receive DIaK in order to improve its ability to validate its data and determine its own health.

The ISHM-Development Toolkit (ISHM-DTK) is an object-oriented environment that enables creation of a model of any complex system (or system-of-systems — SoS) for the ISHM embedded capability. SoS are defined as hierarchical networks of intelligent elements (sensors, components, controllers, processes, sub-systems, systems, etc.).

Integration is established by defining "Intelligent Processes" that represent models of processes that provide the means to check consistency of DIaK across the entire system. Multiple models of varying granularity and fidelity may represent a process, and they may be activated based on context. ISHM-DTK includes communications gateways to read data into the model.

ISHM-DTK allows for modular implementation of ISHM capability with almost total re-use of software. The toolkit also allows incremental implementation of ISHM capability where more and better DIaK is added as these become available or refined in the research and technology community. In order to accommodate legacy elements, such as classical sensors or components, intelligent elements may be virtually implemented in the software, or may use another software environment and/or computer in the network.

This work was done by Jorge Figueroa of Stennis Space Center and Harvey Smith and Jon Morris of Jacobs Technology.

Inquiries concerning this technology should be addressed to the Intellectual Property Manager, Stennis Space Center, (228) 688-1929. Refer to SSC-00255-1, volume and number of this NASA Tech Briefs *issue, and the page number.*