

In the multi-core system, the FTM resides on a single, dedicated core, separate from the cores used by the application. This is done in order to isolate the FTM from application faults and to allow it to swap out any application core for a substitute. The structure of the FTM consists of an interface to a fault tolerant strategy module, a responder module, a fault manager module, an error factory, and an error mapper that determines the severity of the error.

In the present reference implementation, the only fault tolerant strategy implemented is introspection. The introspection code waits for an application node to send an error notification to it. It then uses the error factory to create an error object, and at this time, a severity level is assigned to the error. The introspection code uses its built-in knowledge base to generate a recommended response to the error. Responses might include ignoring the error, logging it, rolling back the application to a previously saved checkpoint, swapping in a new node to replace a bad one, or restarting the application. The original error and recommended response are passed to the top-level fault manager module, which invokes the response. The responder module also notifies the introspection module of the generated response. This provides additional information to the introspection module that it can use in generating its next response. For example, if the responder triggers an application rollback and errors are still occurring, the introspection module may decide to recommend an application restart.

*This work was done by Raphael R. Some, Paul L. Springer, Hans P. Zima, Mark James, and David A. Wagner of Caltech for NASA's Jet Propulsion Laboratory. For more information, contact tiaoffice@jpl.nasa.gov.*

*This software is available for commercial licensing. Please contact Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-47806.*

---

### **DspaceOgreTerrain 3D Terrain Visualization Tool**

DspaceOgreTerrain is an extension to the DspaceOgre 3D visualization tool that supports real-time visualization of various terrain types, including digital elevation maps, planets, and meshes. DspaceOgreTerrain supports creating 3D representations of terrains and placing them in a scene graph. The 3D representations allow for a continuous level of detail, GPU-based ren-

dering, and overlaying graphics like wheel tracks and shadows. It supports reading data from the SimScape terrain-modeling library.

DspaceOgreTerrain solves the problem of displaying the results of simulations that involve very large terrains. In the past, it has been used to visualize simulations of vehicle traverses on Lunar and Martian terrains. These terrains were made up of billions of vertices and would not have been renderable in real-time without using a continuous level of detail rendering technique.

*This work was done by Steven Myint, Abhinandan Jain, and Marc I. Pomerantz 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 Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-47976.*

---

### **Trick Simulation Environment 07**

The Trick Simulation Environment is a generic simulation toolkit used for constructing and running simulations. This release includes a Monte Carlo analysis simulation framework and a data analysis package. It produces all auto documentation in XML. Also, the software is capable of inserting a malfunction at any point during the simulation. Trick 07 adds variable server output options and error messaging and is capable of using and manipulating wide characters for international support. Wide character strings are available as a fundamental type for variables processed by Trick.

A Trick Monte Carlo simulation uses a statistically generated, or predetermined, set of inputs to iteratively drive the simulation. Also, there is a framework in place for optimization and solution finding where developers may iteratively modify the inputs per run based on some analysis of the outputs. The data analysis package is capable of reading data from external simulation packages such as MATLAB and Octave, as well as the common comma-separated values (CSV) format used by Excel, without the use of external converters. The file formats for MATLAB and Octave were obtained from their documentation sets, and Trick maintains generic file readers for each format.

XML tags store the fields in the Trick header comments. For header files,

XML tags for structures and enumerations, and the members within are stored in the auto documentation. For source code files, XML tags for each function and the calling arguments are stored in the auto documentation. When a simulation is built, a top level XML file, which includes all of the header and source code XML auto documentation files, is created in the simulation directory. Trick 07 provides an XML to TeX converter. The converter reads in header and source code XML documentation files and converts the data to TeX labels and tables suitable for inclusion in TeX documents.

A malfunction insertion capability allows users to override the value of any simulation variable, or call a malfunction job, at any time during the simulation. Users may specify conditions, use the return value of a malfunction trigger job, or manually activate a malfunction. The malfunction action may consist of executing a block of input file statements in an action block, setting simulation variable values, call a malfunction job, or turn on/off simulation jobs.

The variable server output options and error messaging capabilities allow the variable server to return data in a tab delimited ASCII format, or in a record-based binary format. The binary record includes information about the type and size of a variable not present in the ASCII format. The binary option is capable of transmitting full C/C++ structures with one request. With this option, error messaging is returnable to client applications. In this software, the variable server may also return time synchronous data, which is gathered and sent at the end of the simulation frame to guarantee data consistency. Also, the Trick 07 variable server is capable of delivering the simulation data to multiple clients using multicast sockets. This allows multiple machines to receive the same data without increasing the computational load on the simulation.

In addition to Linux and MacOSX, Trick 07 now supports three real-time operating systems: QNX, LynxOS, and RedHawk Linux. Each RTOS has unique system calls accessing real-time features such as setting process priorities, processor assignment, and accessing the real-time clock. Trick uses the unique real-time features of each OS.

*This work was done by Alexander S. Lin of Johnson Space Center and John M. Penn, Dan A. Strauss, and Keith Vetter of L-3 Communications Corporation. Further informa-*

## Geometric Reasoning for Automated Planning

An important aspect of mission planning for NASA's operation of the International Space Station is the allocation and management of space for supplies and equipment. The Stowage, Configuration Analysis, and Operations Planning teams collaborate to perform the bulk of that planning. A Geometric Reasoning Engine is developed in a way that can be shared by the teams to optimize item placement in the context of crew planning.

The ISS crew spends (at the time of this writing) a third or more of their time moving supplies and equipment around. Better logistical support and optimized packing could make a significant impact on operational efficiency of the ISS. Currently, computational geometry and motion planning do not focus specifically on the optimized orientation and placement of 3D objects based on multiple distance and containment preferences and constraints.

The software performs reasoning about the manipulation of 3D solid models in order to maximize an objective function based on distance. It optimizes for 3D orientation and placement. Spatial placement optimization is a general problem and can be applied to object packing or asset relocation.

*This work was done by Bradley J. Clement and Russell L. Knight 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 Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-47436.*

## Water Detection Based on Color Variation

This software has been designed to detect water bodies that are out in the open on cross-country terrain at close range (out to 30 meters), using imagery acquired from a stereo pair of color cameras mounted on a terrestrial, unmanned ground vehicle (UGV). This detector exploits the fact that the color variation across water bodies is generally larger and more uniform than that of other naturally occurring types of terrain, such as soil and vegetation. Non-traversable water bodies, such as large puddles, ponds, and lakes, are detected based on color variation, image intensity variance, image intensity gradient, size, and shape.

At ranges beyond 20 meters, water bodies out in the open can be indirectly detected by detecting reflections of the sky below the horizon in color imagery. But at closer range, the color coming out of a water body dominates sky reflections, and the water cue from sky reflections is of marginal use. Since there may be times during UGV autonomous navigation when a water body does not come into a perception system's field of view until it is at close range, the ability to detect water bodies at close range is critical. Factors that influence the perceived color of a water body at close range are the amount and type of sediment in the water, the water's depth, and the angle of incidence to the water body. Developing a single model of the mixture ratio of light reflected off the water surface (to the camera) to light coming out of the water body (to the camera) for all water bodies would be fairly difficult. Instead, this software detects close water bodies based on local

terrain features and the natural, uniform change in color that occurs across the surface from the leading edge to the trailing edge.

From a water body's leading edge to the trailing edge, brightness and saturation levels tend to increase, with saturation content changing at a faster rate than the brightness content. For all the pixels on a water body, a plot of brightness/saturation vs. incidence angle is fairly linear with high slope. Fortunately, this slope tends to be higher for water than other naturally occurring terrain. This phenomenology was exploited here to develop software that detects water bodies at close range. First, candidate water regions are identified in image space by locating regions having low texture. Next, the color changes are evaluated across each candidate water region to locate those consistent with water. Finally, an ellipse fit is performed on remaining candidate water regions and size and aspect ratio filtering is applied to prune regions that geometrically are not likely to be water.

*This work was done by Arturo L. Rankin of Caltech for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).*

*In accordance with Public Law 96-517, the contractor has elected to retain title to this invention. Inquiries concerning rights for its commercial use should be addressed to:*

*Innovative Technology Assets Management*

*JPL*

*Mail Stop 202-233*

*4800 Oak Grove Drive*

*Pasadena, CA 91109-8099*

*E-mail: iaoffice@jpl.nasa.gov*

*Refer to NPO-47088, volume and number of this NASA Tech Briefs issue, and the page number.*