Software

Flightspeed Integral Image Analysis Toolkit

The Flightspeed Integral Image Analysis Toolkit (FIIAT) is a C library that provides image analysis functions in a single, portable package. It provides basic low-level filtering, texture analysis, and subwindow descriptor for applications dealing with image interpretation and object recognition. Designed with spaceflight in mind, it addresses:

- Ease of integration (minimal external dependencies)
- Fast, real-time operation using integer arithmetic where possible (useful for platforms lacking a dedicated floating-point processor)
- Written entirely in C (easily modified)
- "Mostly static" memory allocation
- 8-bit image data

The basic goal of the FIIAT library is to compute meaningful numerical descriptors for images or rectangular image regions. These n-vectors can then be used directly for novelty detection or pattern recognition, or as a feature space for "higher-level" pattern recognition tasks. The library provides routines for leveraging training data to derive descriptors that are most useful for a specific data set. Its runtime algorithms exploit a structure known as the "integral image." This is a caching method that permits fast summation of values within rectangular regions of an image. This integral frame facilitates a wide range of fast image-processing functions.

This toolkit has applicability to a wide range of autonomous image analysis tasks in the space-flight domain, including novelty detection, object and scene classification, target detection for autonomous instrument placement, and science analysis of geomorphology. It makes real-time texture and pattern recognition possible for platforms with severe computational restraints. The software provides an order of magnitude speed increase over alternative software libraries currently in use by the research community.

FIIAT can commercially support intelligent video cameras used in intelligent surveillance. It is also useful for object recognition by robots or other autonomous vehicles.

This work was done by David R. Thompson of Caltech 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-46871.

Work Coordination Engine

The Work Coordination Engine (WCE) is a Java application integrated into the Service Management Database (SMDB), which coordinates the dispatching and monitoring of a work order system. WCE de-queues work orders from SMDB and orchestrates the dispatching of work to a registered set of software worker applications distributed over a set of local, or remote, heterogeneous computing systems. WCE monitors the execution of work orders once dispatched, and accepts the results of the work order by storing to the SMDB persistent store.

The software leverages the use of a relational database, Java Messaging System (JMS), and Web Services using Simple Object Access Protocol (SOAP) technologies to implement an efficient work-order dispatching mechanism capable of coordinating the work of multiple computer servers on various platforms working concurrently on different, or similar, types of data or algorithmic processing. Existing (legacy) applications can be wrapped with a proxy object so that no changes to the application are needed to make them available for integration into the work order system as "workers." WCE automatically reschedules work orders that fail to be executed by one server to a different server if available. From initiation to completion, the system manages the execution state of work orders and workers via a well-defined set of events, states, and actions. It allows for configurable work-order execution timeouts by work-order type.

This innovation eliminates a current processing bottleneck by providing a highly scalable, distributed work-order system used to quickly generate products needed by the Deep Space Network (DSN) to support space flight operations. WCE is driven by asynchronous messages delivered via JMS indicating the availability of new work or workers. It runs completely unattended in support of the lights-out operations concept in the DSN.

This work was done by Silvino Zendejas, Tung Bui, Bach Bui, Shantanu Malhotra, Fannie Chen, Rachel Kim, Christopher Allen, Ivy Luong, and George Chang of Caltech and Syed Sadaqathulla of Raytheon 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-45014.

Multi-Mission Automated Task Invocation Subsystem

Multi-Mission Automated Task Invocation Subsystem (MATIS) is software that establishes a distributed data-processing framework for automated generation of instrument data products from a spacecraft mission. Each mission may set up a set of MATIS servers for processing its data products. MATIS embodies lessons learned in experience with prior instrument-data-product-generation software.

MATIS is an event-driven workflow manager that interprets project-specific, user-defined rules for managing processes. It executes programs in response to specific events under specific conditions according to the rules. Because requirements of different missions are too diverse to be satisfied by one program, MATIS accommodates plug-in programs. MATIS is flexible in that users can control such processing parameters as how many pipelines to run and on which computing machines to run them.

MATIS has a fail-safe capability. At each step, MATIS captures and retains pertinent information needed to complete the step and start the next step. In the event of a restart, this information is retrieved so that processing can be resumed appropriately.

At this writing, it is planned to develop a graphical user interface (GUI) for monitoring and controlling a product-generation engine in MATIS. The GUI would enable users to schedule multiple processes and manage the data products produced in the processes. Although MATIS was initially designed for instrument data product generation, the architecture does not preclude it from being used for different applications. It is planned that the MATIS team members will provide a set of application