Several unique challenges influenced design decisions for automatic image analysis. First, onboard processing is limited in spaceflight applications. Avionics computers must satisfy strict radiation and energy constraints, and their resources are shared between continuous autonomous control and data processing. Computational constraints mandate a simple approach to image analysis in which statistical properties of the image serve as proxies for the actual content.

A major challenge is the diversity of surface features an aerobot might encounter. An aerobot would be in constant motion but difficult to control due to unpredictable atmospheric currents. It would be difficult to schedule image targets in advance or to anticipate the features of interest that will appear. This favors an "unsupervised" approach that makes few assumptions about image content but instead discovers interesting and representative samples based on the intrinsic properties of the data. Clustering is one common unsupervised approach; it classifies a dataset into discrete categories of items with similar properties.

Image features can be considered to fall into one of four groups, or themes, based on the properties they describe. These are color, edge, frequency, and time. The edge and frequency features correlate with image texture, color captures basic color statistics, and time describes the temporal order in which the images were collected.

The main feature of this innovation is the use of clustering/machine learning methods to structure data for prioritization, mapping, and downlink. The effectiveness of clustering rests on the quality of the feature vectors describing each set of data. Features that are redundant, or have little variance, will reduce the effectiveness of clustering. Dimensionality reduction techniques such as principal component analysis (PCA) can transform a high-dimensional feature space into a lower-dimensional space where the new, uncorrelated features have heightened variance. Ideal clusterings contain compact clusters that are spread far apart from one another.

This work was done by Steve A. Chien, David Hayden, David R. Thompson, and Rebecca Castano of Caltech for NASA's Jet Propulsion Laboratory. For more information, contact iaoffice@jpl.nasa.gov. NPO-47534

Monitoring and Acquisition Real-time System (MARS)

Marshall Space Flight Center, Alabama

MARS is a graphical user interface (GUI) written in MATLAB and Java, allowing the user to configure and control the Scalable Parallel Architecture for Real-Time Acquisition and Analysis (SPARTAA) data acquisition system. SPARTAA not only acquires data, but also allows for complex algorithms to be applied to the acquired data in real time. The MARS client allows the user to set up and configure all settings regarding the data channels attached to the system, as well as have complete control over starting and stopping data acquisition. It provides a unique "Test" programming environment, allowing the user to create tests consisting of a series of alarms, each of which contains any number of data channels. Each alarm is configured with a particular algorithm, determining the type of processing that will be applied on each data channel and tested against a defined threshold. Tests can be uploaded to SPARTAA, thereby teaching it how to process the data.

MARS was developed as a front-end GUI for setup, control, and plotting of data from SPARTAA. The system was designed to monitor spectral components in real time from instrumentation located on high-speed rotational hardware (primarily high-pressure turbopumps), and to issue cut commands to a facility if preset levels were violated. However, the system is not limited to rotational hardware, and can be used to monitor any level of frequency information from a myriad of instrumented test hardware. The control software allows the user to configure the system easily to support testing of various configurations with multiple alarms, voting logic, and sensor validation,

The uniqueness of MARS is in its capability to be adaptable easily to many test configurations. Test hardware measurement limits (i.e. vibration, pressure, temperature, etc.) can be predetermined, and MARS can be used to set up and support quickly any test configuration. Multiple alarms with various timings can be configured within minutes, as opposed to previous software modifications. MARS sends and receives protocols via TCP/IP, which allows for quick integration into almost any test environment. The use of MATLAB and Java as the programming languages allows for developers to integrate the software across multiple operating platforms.

This work was done by Corbin Holland of Marshall Space Flight Center. For more information, contact Sammy Nabors, MSFC Commercialization Assistance Lead, at sammy.a.nabors@nasa.gov. Refer to MFS-32905-1.