# Software

## *In Situ* Mosaic Brightness Correction

*In situ* missions typically have pointable, mast-mounted cameras, which are capable of taking panoramic mosaics comprised of many individual frames. These frames are mosaicked together. While the mosaic software applies radiometric correction to the images, in many cases brightness/contrast seams still exist between frames. This is largely due to errors in the radiometric correction, and the absence of correction for photometric effects in the mosaic processing chain. The software analyzes the overlaps between adjacent frames in the mosaic and determines correction factors for each image in an attempt to reduce or eliminate these brightness seams.

Two related methods of correcting brightness differences at seams between frames in a mosaic of *in situ* images work on the same general principle. The overlapping areas between adjacent frames in a mosaic are analyzed, and statistics are gathered. These statistics are then used in a bundle-adjustment style procedure to derive correction parameters for each image that minimize the brightness seams across the mosaic.

The older method consists of two programs: marsint, which gathers overlap statistics, and marsbias, which determines correction parameters. The newer system adds additional capabilities, including simultaneous brightness and contrast correction, better overlap statistics, improved image labels, and standard XML file formats. The overlap analysis functionality is embedded in the mosaic program marsmap, while the correction parameters are determined by the program marsbrt.

In both cases, the correction parameters are input to the mosaic program of interest (marsmap, marsmos, marsmcauley) to be applied to the mosaic. Correction parameters are constant additive or multiplicative factors applied to the entire input image; no nonlinear corrections are applied.

The software is part of the OPGS (Operational Product Generation Subsystem) software suite. While the algorithms behind this suite are not particularly unique, what makes the programs useful is their integration into the larger *in situ* image processing system via the PIG library and the mosaic programs. They work directly with space *in situ* data, understanding the appropriate image metadata fields and updating them properly.

*This work was done by Robert G. Deen and Jean J. Lorre of Caltech for NASA's Jet Propulsion Laboratory. For more information, contact iaoffice@jpl.nasa.gov. NPO-47726*

## Simplex GPS and InSAR Inversion Software

Changes in the shape of the Earth's surface can be routinely measured with precisions better than centimeters. Processes below the surface often drive these changes and as a result, investigators require models with inversion methods to characterize the sources. Simplex inverts any combination of GPS (global positioning system), UAVSAR (uninhabited aerial vehicle synthetic aperture radar), and InSAR (interferometric synthetic aperture radar) data simultaneously for elastic response from fault and fluid motions. It can be used to solve for multiple faults and parameters, all of which can be specified or allowed to vary. The software can be used to study long-term tectonic motions and the faults responsible for those motions, or can be used to invert for co-seismic slip from earthquakes. Solutions involving estimation of fault motion and changes in fluid reservoirs such as magma or water are possible. Any arbitrary number of faults or parameters can be considered.

Simplex specifically solves for any of location, geometry, fault slip, and expansion/contraction of a single or multiple faults. It inverts GPS and InSAR data for elastic dislocations in a half-space. Slip parameters include strike slip, dip slip, and tensile dislocations. It includes a map interface for both setting up the models and viewing the results. Results, including faults, and observed, computed, and residual displacements, are output in text format, a map interface, and can be exported to KML. The software interfaces with the QuakeTables database allowing a user to select existing fault parameters or data. Simplex can be accessed through the QuakeSim portal graphical user interface or run from a UNIX command line.

*This work was done by Andrea Donnellan, Jay W. Parker, and Gregory A. Lyzenga of Caltech, and Marlon E. Pierce of Indiana University for NASA's Jet Propulsion Laboratory. For more information, contact iaoffice @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-48233.*

## Virtual Machine Language 2.1

VML (Virtual Machine Language) is an advanced computing environment that allows spacecraft to operate using mechanisms ranging from simple, time-oriented sequencing to advanced, multi-component reactive systems.

VML has developed in four evolutionary stages. VML 0 is a core execution capability providing multi-threaded command execution, integer data types, and rudimentary branching. VML 1 added named parameterized procedures, extensive polymorphism, data typing, branching, looping issuance of commands using run-time parameters, and named global variables. VML 2 added for loops, data verification, telemetry reaction, and an open flight adaptation architecture. VML 2.1 contains major advances in control flow capabilities for executable state machines.

On the resource requirements front, VML 2.1 features a reduced memory footprint in order to fit more capability into modestly sized flight processors, and endian-neutral data access for compatibility with Intel little-endian processors. Sequence packaging has been improved with object-oriented programming constructs and the use of implicit (rather than explicit) time tags on statements. Sequence event detection has been significantly enhanced with multi-variable waiting, which allows a sequence to detect and react to conditions defined by complex expressions with multiple global variables. This multi-variable waiting serves as the basis for implementing parallel rule checking, which in turn, makes possible executable state machines.

The new state machine feature in VML 2.1 allows the creation of sophisticated autonomous reactive systems without the need to develop expensive flight software. Users specify named states and transitions, along with the truth conditions required, before taking transitions. Transi-