$[b/(b+1)]$ binary convolutional code (or a short block code) with maximum free Hamming distance (or minimum distance). The interleaver (π) permutes the output of the outer encoder. The interleaved data enter the inner encoder, which implements a rate-(m/m) [rate-1] recursive convolutional code. The m output bits are then mapped to one symbol that belongs to a 2^m-level modulation. Because the inner code does not have redundancy, it is useless by itself; however, the combination of the inner and outer codes with the interleaver results in very powerful code. For MQAM where $M = N^2$, further reduction in complexity is possible. This can be done by assigning the $m = \log_2 N$ output bits of the inner encoder alternately to the in-phase and quadrature components of N^2QAM modulation. In this case, the bit-rate-to-bandwidth ratio will be

$$2bm/(b+1).$$

The advantage of this generic design can be made more apparent by citing an example of $b = 3$ for 16QAM, for which $m = 2$. In this example, the number of transitions per state of the inner TCM is only 4, which is only 1/32 of the corresponding number for the previous case.

This work was done by Dariush Divsalar, Sam Dolinar, and Fabrizio Pollara 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:

Intellectual Property group
JPL
Mail Stop 202-233
4800 Oak Grove Drive
Pasadena, CA 91109
(818) 354-2240

Refer to NPO-20878, volume and number of this NASA Tech Briefs *issue, and the page number.*

Enhanced Software for Scheduling Space-Shuttle Processing

Prototype software has been upgraded.

John F. Kennedy Space Center, Florida

The Ground Processing Scheduling System (GPSS) computer program is used to develop streamlined schedules for the inspection, repair, and refurbishment of space shuttles at Kennedy Space Center. A scheduling computer program is needed because space-shuttle processing is complex and it is frequently necessary to modify schedules to accommodate unanticipated events, unavailability of specialized personnel, unexpected delays, and the need to repair newly discovered defects. GPSS implements constraint-based scheduling algorithms and provides an interactive scheduling software environment. In response to inputs, GPSS can respond with schedules that are optimized in the sense that they contain minimal violations of constraints while supporting the most effective and efficient utilization of space-shuttle ground processing resources.

The present version of GPSS is a product of re-engineering of a prototype version. While the prototype version proved to be valuable and versatile as a scheduling software tool during the first five years, it was characterized by design and algorithmic deficiencies that affected schedule revisions, query capability, task movement, report capability, and overall interface complexity. In addition, the lack of documentation gave rise to difficulties in maintenance and limited both enhanceability and portability.

The goal of the GPSS re-engineering project was to upgrade the prototype into a flexible system that supports multiple-flow, multiple-site scheduling and that retains the strengths of the prototype while incorporating improvements in maintainability, enhanceability, and portability. The major enhancements were the following:

- The implementation of container objects (e.g., lists and maps) was made more efficient by use of the C++ Standard Template Library (STL).
- Improvements in the management of schedule network objects were made. An embedded schedule-data configuration-management subsystem, similar to systems used for software configuration management, was built. This subsystem accommodates multiple versions and revisions of each schedule, including direct descendants and branches. It also implements a concept of user sessions that enables each user to maintain multiple current instances of the same schedule and full schedule data files with sizes of the order of 1MB.
- Improvements in calendar operations were made. The original implementation required the full, time-series expansion of all calendars, giving rise to a large memory overhead. Furthermore, some calendar features (e.g., holidays), were "hard-coded." In the re-engineering, calendar memory requirements were reduced by providing for all calendar calculations to be performed in real time and by removing all hard-coded elements.
- Re-engineering of a robust query subsystem was perhaps the most challenging aspect of the project. The prototype utilized a Prolog-like query language that was scanned, parsed, and executed in a C program. The query code was problematic and difficult to understand. The re-engineering involved the building of a real (but similar) query language, utilizing the FLEX language and the Bison program to define a scanner and parser that includes all elements of logical inference (for example, AND, OR, and NOT) as well as full capability for building and incorporating user-customizable queries.
- An improved report architecture was developed. The prototype featured a significant number of hard-coded user options, and too little care was taken initially to develop a consistent but flexible report architecture. The re-engineering of the affected software components involved design around a new report class that contains attributes that describe the class of objects (e.g., tasks) represented in a report, the presentation style (e.g., Gantt chart or tabulation), and the time frame of the report. All report definitions are saved in files that the user can edit to customize reports.
- Several improvements in algorithms were made to solve backward-movement problems, provide a more robust implementation of achievers, and improve memory management through the use of smart pointers and "lazy load" of persistent data. Also included

is an updated implementation of an object-oriented callback system to the Motif widget set.

The benefits of the re-engineered version of GPSS hinge on the object-oriented approach. The use of STL and the improvements in schedule and query operations are incorporated in C++ libraries that may prove useful on succeeding projects. The rewriting of software in C++ increases portability. For the users, every effort was made in the re-engineering to maximize flexibility and improve upon the intuitive nature of the interface without sacrificing any of the capabilities that made the prototype successful.

This work was done by Joseph A. Barretta, Earl P. Johnson, Rocky R. Bierman, Juan Blanco, Kathleen Boaz, Lisa A. Stotz, Michael Clark, George Lebovitz, Kenneth J. Lotti, James M. Moody, Tony K. Nguyen, Kenneth A. Peterson, Susan Sargent, Karma Shaw, Mack D. Stoner, Deborah S. Stowell, Daniel A. Young, and James H. Tulley, Jr., of United Space Alliance for **Kennedy Space Center**. *For further information, contact the Kennedy Commercial Technology Office at 321-867-8130.*
KSC-12043

Bayesian-Augmented Identification of Stars in a Narrow View

An adaptive threshold guides acceptance or rejection of a tentative identification.

NASA's Jet Propulsion Laboratory, Pasadena, California

An algorithm for the identification of stars from a charge-coupled-device (CCD) image of a star field has been extended for use with narrower field-of-view images. Previously, the algorithm had been shown to be effective at a field of view of 8°. This work augments the earlier algorithm using Bayesian decision theory. The new algorithm is shown to be capable of effective star identification down to a field of view of 2°. The algorithm was developed for use in estimating the attitude of a spacecraft and could be used on Earth to help in the identification of stars and other celestial objects for astronomical observations.

The present algorithm is one of several that seek matches between (1) imaged star fields and (2) portions of the sky, with angular dimensions equal to those of the imaged star fields, in a catalog of stars in a known reference frame. Previously developed star-identification algorithms are not suitable for fields of view only 2° wide. The present algorithm is based partly on one such prior algorithm, called the "grid algorithm," that has shown promise for identifying stars in fields of view 8° wide. To make it possible to identify stars in fields of view down to 2° with acceptably low probabilities of error, the grid algorithm has been extended by incorporating Bayesian decision theory.

For the special purpose of the grid algorithm, the term "pattern" denotes a grid representation of the relative positions of stars in a field of view. Each star is deemed to be located within one of the cells of a square grid that spans either the field of view of the CCD image or a candidate star-catalog field of the same angular dimensions. The portion of the grid algorithm that generates a pattern comprises the following steps (see figure):

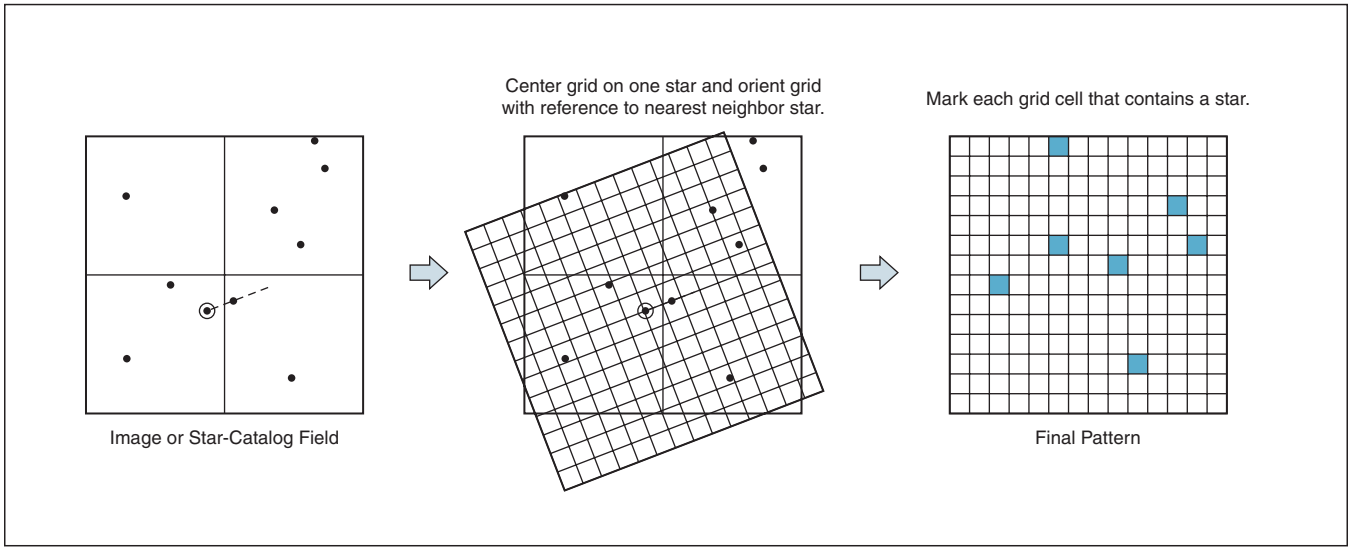
1. Choose a star from the CCD image or the applicable field of view in the star catalog to be the center star.
2. Decide which star is the neighbor star. The neighbor star is deemed to be the star nearest to the center star outside a buffer radius of *br* pixels. The value of *br* is chosen on the basis of experience.
3. Center a grid of *g* rows and *g* columns on the center star, and orient the grid such that a horizontal vector from the center to the right edge passes through the neighbor star. Like *br*, the value of *g* is chosen on the basis of experience.
4. Derive a pattern, a g^2-element bit vector

$$V[0 \ldots g^2 - 1]$$

such that if grid cell (i,j) contains a star, then

$$V[jg + i] = 1$$

The vector element corresponding to any grid cell which does not contain a star is given the value 0. The dot product



A **Grid Pattern** is created from either a CCD image of stars or star-catalog data for a field of view of the same angular dimensions as those of the CCD image.