Improved Compression of Wavelet-Transformed Images

Code parameters are selected adaptively to achieve high compression performance.

NASA's Jet Propulsion Laboratory, Pasadena, California

A recently developed data-compression method is an adaptive technique for coding quantized wavelet-transformed data, nominally as part of a complete image-data compressor. Unlike some other approaches, this method admits a simple implementation and does not rely on the use of large code tables.

A common data compression approach, particularly for images, is to perform a wavelet transform on the input data, and then losslessly compress a quantized version of the wavelet-transformed data. Under this compression approach, it is common for the quantized data to include long sequences, or "runs," of zeros.

The new coding method uses prefix-free codes for the nonnegative integers as part of an adaptive algorithm for compressing the quantized wavelet-transformed data by run-length coding. In the form of run-length coding used here, the data sequence to be encoded is parsed into strings consisting of some number (possibly 0) of zeros, followed by a nonzero value. The nonzero value and the length of the run of zeros are encoded. For a data stream that contains a sufficiently high frequency of zeros, this method is known to be more effective than using a single variable length code to encode each symbol. The specific prefix-free codes used are from two classes of variable-length codes: a class known as Golomb codes, and a class known as exponential-Golomb codes. The codes within each class are indexed by a single integer parameter.

The present method uses exponential-Golomb codes for the lengths of the runs of zeros, and Golomb codes for the nonzero values. The code parameters within each code class are determined adaptively on the fly as compression proceeds, on the basis of statistics from previously encoded values. In particular, a simple adaptive method has been devised to select the parameter identifying the particular exponential-Golomb code to use. The method tracks the average number of bits used to encode recent runlengths, and takes the difference between this average length and the code parameter. When this difference falls outside a fixed range, the code parameter is updated (increased or decreased). The Golomb code parameter is selected based on the average magnitude of recently encoded nonzero samples.

The coding method requires no floating-point operations, and more readily adapts to local statistics than other methods. The method can also accommodate arbitrarily large input values and arbitrarily long runs of zeros. In practice, this means that changes in the dynamic range or size of the input data set would not require a change to the compressor.

The algorithm has been tested in computational experiments on test images. A comparison with a previously developed algorithm that uses large code tables (generated via Huffman coding on training data) suggests that the data-compression effectiveness of the present algorithm is comparable to the best performance achievable by the previously developed algorithm.

This work was done by Aaron Kiely and Matthew Klimesh of Caltech for **NASA's Jet Propulsion Laboratory**. *Further information is contained in a TSP (see page 1).*

The software used in this innovation is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (818) 393-2827. Refer to NPO-40490.

NASA Interactive Forms Type Interface — NIFTI

A flexible database query, update, modify, and delete tool provides an easy interface to Oracle forms.

Lyndon B. Johnson Space Center, Houston, Texas

A flexible database query, update, modify, and delete tool was developed that provides an easy interface to Oracle forms. This tool — the NASA interactive forms type interface, or NIFTI — features on-the-fly forms creation, forms sharing among users, the capability to query the database from user-entered criteria on forms, traversal of query results, an ability to generate tab-delimited reports, viewing and downloading of reports to the user's workstation, and a hypertext-based help system. NIFTI is a very powerful *ad hoc* query tool that was developed using C++, X-Windows by a Motif application framework. A unique tool, NIFTI's capabilities appear in no other known commercial-off-the-shelf (COTS) tool, because NIFTI, which can be launched from the user's desktop, is a simple yet very powerful tool with a highly intuitive, easy-to-use graphical user interface (GUI) that will expedite the creation of database query/update forms. NIFTI, therefore, can be used in NASA's International Space Station (ISS) as well as within government and industry — indeed by all users of the widely disseminated Oracle base. And it will provide significant cost savings in the areas of user training and scalability while advancing the art over current COTS browsers.

No COTS browser performs all the functions NIFTI does, and NIFTI is easier to use. NIFTI's cost savings are very significant considering the very large database with which it is used and the large user community with varying data requirements it will support. Its ease of use means that personnel unfamiliar with databases (e.g., managers, supervisors, clerks, and others) can develop their own personal reports. For NASA, a tool such as NIFTI was needed to query, update, modify, and make deletions within the ISS vehicle master database (VMDB), a repository of engineering data that includes an indentured parts list and associated resource data (power, thermal, volume, weight, and the like). Since the VMDB is used both as a collection point for data and as a common repository for engineering, integration, and operations teams, a tool such as NIFTI had to be designed that could expedite the creation of database query/update forms which could then be shared among users.

The present state of the art with COTS browsers means a user must have completed a form to access data in the VMDB. This means a user brings his/her need to the attention of management. The management then brings it to the attention of vehicle data management. The importance of data access is measured against other competing needs for database access, and the required access is eventually deemed sufficiently important to allocate requirements for the VMDB team. This requirement is scheduled for satisfaction at some future release of the VMDB and is assigned to a VMDB team developer. The developer meets with the requester to hammer out requirements for the form. The form is then implemented with Oracle and is captured within the software configuration management system for the VMDB. There the software will likely exist and continue to be maintained for at least the next decade. If the user requires any changes to the form afterwards, the lengthy process detailed above must be repeated.

NIFTI truncates this process. With NIFTI, a user selects the table he/she wants to access and then selects columns from that table. The field is sized to the user's specifications, and labels and titles are added. A search string is entered on the user's form, including a wildcard, and the user presses the query button. The data needed are now there. If the user is responsible for updating or inserting in the database, he/she corrects the data on the form or enters new data and selects the update or insert button. Since the form has been saved in the database, it has also been saved as part of the database backup; that is, it is not part of the configuration management process. This means that should the user want to add another column or perform a join with another table, he/she can do this as well. The user can then share created forms with another VMDB user simply by pressing a toggle button and saving. However, should the user not wish to share his/her form, this can be done too by marking the form private. Private forms, which are viewable only by the user, are unlike public forms, which can be viewed by all NIFTI users.

NIFTI is an extremely flexible and reliable tool that can be used in place of Oracle forms. As an X-Windows-based application, NIFTI can run on various platforms. At the time of reporting this information, NIFTI was running on a Sun SPARC workstation at Johnson Space Center.

This work was done by Bobby Jain and Bill Morris of Barrios Technology for **Johnson Space Center**. *For further information, contact:*

Barrios Technology, Inc.
2525 Bay Area Blvd., Suite 300
Houston, TX 77058-1556
Phone: (281) 280-1900
Fax: (281) 280-1901

Refer to MSC-22914, volume and number of this NASA Tech Briefs *issue, and the page number.*

Predicting Numbers of Problems in Development of Software

Lyndon B. Johnson Space Center, Houston, Texas

A method has been formulated to enable prediction of the amount of work that remains to be performed in developing flight software for a spacecraft. The basic concept embodied in the method is that of using an idealized curve (specifically, the Weibull function) to interpolate from (1) the numbers of problems discovered thus far to (2) a goal of discovering no new problems after launch (or six months into the future for software already in use in orbit). The steps of the method can be summarized as follows:

1. Take raw data in the form of problem reports (PRs), including the dates on which they are generated.
2. Remove, from the data collection, PRs that are subsequently withdrawn or to which no response is required.
3. Count the numbers of PRs created in 1-week periods and the running total number of PRs each week.
4. Perform the interpolation by making a least-squares fit of the Weibull function to (a) the cumulative distribution of PRs gathered thus far and (b) the goal of no more PRs after the currently anticipated launch date. The interpolation and the anticipated launch date are subject to iterative re-estimation.

This work was done by Charles H. Simonds of Lockheed Martin Corp. for **Johnson Space Center**. *For further information, contact the Johnson Innovative Partnerships Office at (281) 483-3809. MSC-23532*