Efficient Algorithm for Rectangular Spiral Search

The search pattern is automatically expanded as needed.

NASA's Jet Propulsion Laboratory, Pasadena, California

An algorithm generates grid coordinates for a computationally efficient spiral search pattern covering an uncertain rectangular area spanned by a coordinate grid. The algorithm does not require that the grid be fixed; the algorithm can search indefinitely, expanding the grid and spiral, as needed, until the target of the search is found. The algorithm also does not require memory of coordinates of previous points on the spiral to generate the current point on the spiral.

The search is started at a point (more precisely, in a grid cell), denoted the center of the spiral, that has been chosen previously as the point most likely to coincide with the target. The search is to be performed on a grid of $n \times m$ cells, where $m > n$ and $a \equiv m - n$ is denoted the rectangular excess of the search pattern (see Figure 1). The spiral search is performed in steps numbered simply 1, 2, 3..., and t represents the number of the current step.

The inputs to the algorithm are t and the rectangular excess that is either specified in advance or calculated from the rectangular grid used to span the initial search area. The output of the algorithm is the pair of integer coordinates (i,j) of the current point on the spiral with respect to the center of the spiral. Figure 2 presents, as an example, the first 15 steps of the spiral generated by this algorithm for a search that starts at a point in a 3×5 rectangle.

This work was done by Paul Brugarolas and William Breckenridge 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-42057.

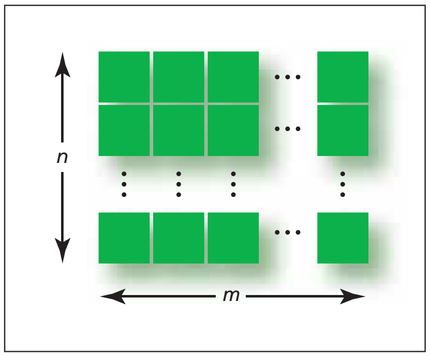

Figure 1. A **Rectangular Grid** of $n \times m$ cells is overlaid on the area to be searched.

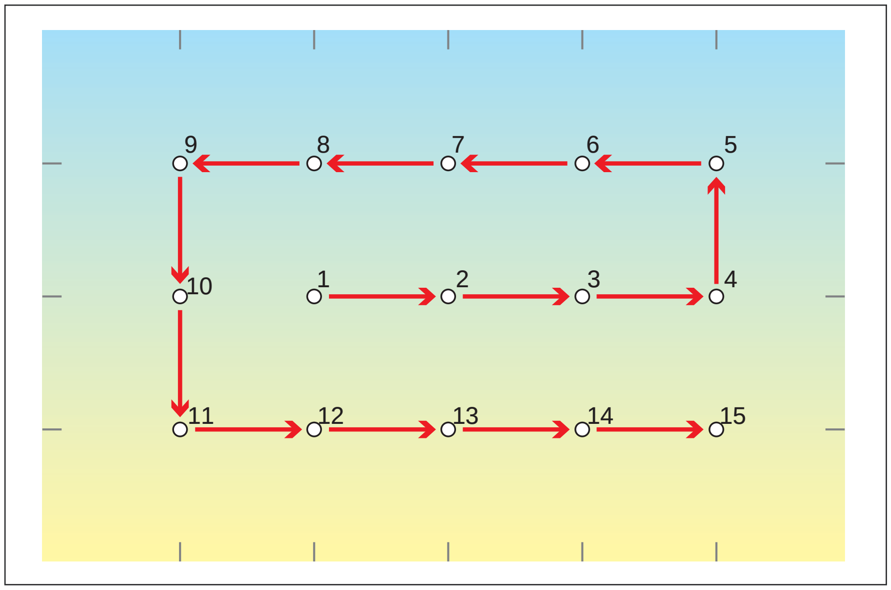

Figure 2. A **Spiral Search Pattern** was generated in an area initially overlaid with a 3×5 grid.

Algorithm-Based Fault Tolerance Integrated With Replication

NASA's Jet Propulsion Laboratory, Pasadena, California

In a proposed approach to programming and utilization of commercial off-the-shelf computing equipment, a combination of algorithm-based fault tolerance (ABFT) and replication would be utilized to obtain high degrees of fault tolerance without incurring excessive costs. The basic idea of the proposed approach is to integrate ABFT with replication such that the algorithmic portions of computations would be protected by ABFT, and the logical portions by replication.

ABFT is an extremely efficient, inexpensive, high-coverage technique for detecting and mitigating faults in computer systems used for algorithmic computations, but does not protect against errors in logical operations surrounding algorithms. Replication is a generally applicable, high-coverage technique for protecting general computations from faults, but is inefficient and costly because it requires additional computation time or additional computational circuitry (and, hence, additional mass and power). The goal of the proposed integration of ABFT with replication is to optimize the fault-tolerance aspect of the design of a computing system by using the less-efficient, more-expensive technique to protect only those computations that cannot be protected by the more-efficient, less-expensive technique. It would not be necessary to address the fault-tolerance issue explicitly in writing an application program to be executed in such a system. Instead, ABFT and replication would be managed by middleware containing hooks.

This work was done by Raphael Some and David Rennels 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-43842.

Targeting and Localization for Mars Rover Operations

NASA's Jet Propulsion Laboratory, Pasadena, California

A design and a partially developed application framework were presented for improving localization and targeting for surface spacecraft. The program has value for the Mars Science Laboratory mission, and has been delivered to support the Mars Exploration Rovers as part of the latest version of the Maestro science planning tool. It also has applications for future missions involving either surface-based or low-altitude atmospheric robotic vehicles.

The targeting and localization solutions solve the problem of how to integrate localization estimate updates into operational planning tools, operational data product generalizations, and flight software by adding expanded flexibility to flight software, the operations data product pipeline, and operations planning tools based on coordinate frame updates during a planning cycle. When acquiring points of interest (targets) for the rover, instead of using a temporal method for reusing previously acquired targets, this system uses a spatial method to avoid tedious and repetitive target re-designation needed to keep target relevance accurate. Instead of creating a target that is reusable only for a sol (Martian day), the target is defined in a way to make it reusable for a planning position (the vehicle position indicated by a Site and Drive index pair) from which the vehicle will begin a command cycle.

This work was done by Mark W. Powell, Thomas Crockett, Jason M. Fox, Joseph C. Joswig, Jeffrey S. Norris, and Kenneth J. Rabe of 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-43847.

Terrain-Adaptive Navigation Architecture

NASA's Jet Propulsion Laboratory, Pasadena, California

A navigation system designed for a Mars rover has been designed to deal with rough terrain and/or potential slip when evaluating and executing paths. The system also can be used for any off-road, autonomous vehicles. The system uses more sophisticated terrain analysis, but also converges to computational complexity similar to that of currently deployed navigation systems when the terrain is benign. The system consists of technologies that have been developed, integrated, and tested onboard research rovers in Mars analog terrains, including goodness maps and terrain triage, terrain classification, remote slip prediction, path planning, high-fidelity traversability analysis (HFTA), and slip-compensated path following.

The system enables vehicles to autonomously navigate different terrain challenges including dry river channel systems, putative shorelines, and gullies emanating from canyon walls. Several of the technologies within this innovation increase the navigation system's capabilities compared to earlier rover navigation algorithms.

This work was done by Daniel M. Helmick, Anelia Angelova, Larry H. Matthies, and Daniel M. Helmick 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- 44588.

Self-Adjusting Hash Tables for Embedded Flight Applications

NASA's Jet Propulsion Laboratory, Pasadena, California

A common practice in computer science to associate a value with a key is to use a class of algorithms called a hash-table. These algorithms enable rapid storage and retrieval of values based upon a key. This approach assumes that many keys will need to be stored immediately. A new set of hash-table algorithms optimally uses system resources to ideally represent keys and values in memory such that the information can be stored and retrieved with a minimal amount of time and space. These hash-tables support the efficient addition of new entries. Also, for large data sets, the look-up time for large data-set searches is independent of the number of items stored, i.e., O(1), provided that the chance of collision is low.

Like arrays, hash-tables provide constant time O(1) look-up on average, regardless of the number of items in the table. However, the rare worst-case look-up time can be as bad as $O(n)$. Compared to other associative array data structures, hash-tables are most useful when large numbers of records are to be stored, especially if the size of the data set can be predicted.