View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by NASA Technical Reports Server

The application layer provides application programs that a robot can execute. Examples of application programs include those needed to perform such prescribed maneuvers as avoiding obstacles while moving from a specified starting point to a specified goal point or turning a robot in place through a specified azimuthal angle. Each robot is provided with application software representing its own unique set of commands. The software establishes a graphical user interface (GUI) for exchanging command information with external computing systems. Via the GUI and its supporting interface software, a user can select and assemble, from the aforementioned set, commands appropriate to the task at hand and send the commands to the robot for execution. System software that interacts with the R4SA software at all three levels establishes a synchronized control environment.

The R4SA software features two modes of execution: before real time (BRT) and real time (RT). In the BRT mode, a text configuration file is read in (each robot has its own unique file) and then device-driver-layer, device-layer, and application-layer initialization functions are executed. If execution is successful, then the system jumps into the RT mode, in which the system is ready to receive and execute commands.

One goal in developing the R4SA architecture was to provide one computer code for many robots. The unique executable code for each robot is built by use of a configuration feature file. The set of features for a given robot is selected from a feature database on the basis of the hardware and mechanical capabilities of that robot. Recompilation of code is straightforward: modifications can readily be performed in the field by use of simple laptop-computer development and debugging software tools.

This work was done by Hrand Aghazarian, Eric Baumgartner, and Michael Garrett 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 (626) 395-2322. Refer to NPO-41796.

R4SA for Controlling Robots

NASA's Jet Propulsion Laboratory, Pasadena, California

The R4SA GUI mentioned in the immediately preceding article is a user-friendly interface for controlling one or more robot(s). This GUI makes it possible to perform meaningful real-time field experiments and research in robotics at an unmatched level of fidelity, within minutes of setup. It provides such powerful graphing modes as that of a digitizing oscilloscope that displays up to 250 variables at rates between 1 and 200 Hz. This GUI can be configured as multiple intuitive interfaces for acquisition of data, command, and control to enable rapid testing of subsystems or an entire robot system while simultaneously performing analysis of data.

The R4SA software establishes an intuitive component-based design environment that can be easily reconfigured for any robotic platform by creating or editing setup configuration files. The R4SA GUI enables event-driven and conditional sequencing similar to those of Mars Exploration Rover (MER) operations. It has been certified as part of the MER ground support equipment and, therefore, is allowed to be utilized in conjunction with MER flight hardware. The R4SA GUI could also be adapted to use in embedded computing systems, other than that of the MER, for commanding and real-time analysis of data.

This work was done by Hrand Aghazarian 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 (626) 395-2322. Refer to NPO-41797.

Bio-Inspired Neural Model for Learning Dynamic Models

This model could be a basis for fast speech- and image-recognition computers.

NASA's Jet Propulsion Laboratory, Pasadena, California

A neural-network mathematical model that, relative to prior such models, places greater emphasis on some of the temporal aspects of real neural physical processes, has been proposed as a basis for massively parallel, distributed algorithms that learn dynamic models of possibly complex external processes by means of learning rules that are local in space and time. The algorithms could be made to perform such functions as recognition and prediction of words in speech and of objects depicted in video images. The approach embodied in this model is said to be "hardware-friendly" in the following sense: The algorithms would be amenable to execution by special-purpose computers implemented as very-large-scale integrated (VLSI) circuits that would operate at relatively high speeds and low power demands.

It is necessary to present a large amount of background information to give meaning to a brief summary of the present neural-network model:

- A dynamic model to be learned by the present neural-network model is of a type denoted an internal model or predictor. In simplest terms, an internal model is a set of equations that predicts future measurements on the basis of past and current ones. Internal models have been used in controlling industrial plants and machines (including robots).
- One of the conclusions drawn from Pavlov's famous experiments was the observation that reinforcers of learning (basically, rewards and punishments) become progressively less efficient for causing adaptation of

behavior as their predictability grows during the course of learning. The difference between the actual occurrence and the prediction of the reinforcer is denoted as the reinforcement prediction error signal. In what is known in the art as the temporal-difference model (TD model) of Pavlovian learning, the reinforcement prediction error signal is used to learn a reinforcement prediction signal. The desired reinforcement prediction signal is defined as a weighted sum of future reinforcement signals wherein reinforcements are progressively discounted with increasing time into the future. The reinforcement prediction error signal progressively decreases during learning as the reinforcement prediction signal becomes more similar to the desired reinforcement prediction signal.

- Algorithms based on the TD model ("TD algorithms") have been analyzed and shown to be implementations of a variant of dynamic programming. Machine-learning studies have shown that TD algorithms are powerful means of solving reinforcement learning problems that involve delayed reinforcement. Examples of such problems include board games, which involve delayed rewards (winning) or punishments (losing).
- Mid-brain dopamine neurons are so named because they modulate the levels of activity of other neurons by means of the neurotransmitter chemical dopamine. The cell bodies of dopamine neurons are located in the brain stem and their axons project to many brain areas. Activities of dopamine neurons have been found to be strikingly similar to the prediction error signals of the TD model.
- Real neural signals include spikelike pulses, the times of occurrence of which are significant. The term "biological spike coding" denotes, essentially, temporal labeling of such pulses in real neurons or in a neural-network model.

This concludes the background information.

The present neural-network model incorporates biological spike coding along with some basic principles of the learning by synapses in the cortex of the human brain. According to the learning rule of the model, synaptic weights are adapted when pre- and postsynaptic spikes occur within short time windows. In simplified terms, for a given synapse and time window, the synaptic strength is increased in the long term if the presynaptic spike precedes the postsynaptic spike or is decreased in the long term if the presynaptic spike follows the postsynaptic spike. This learning rule has been shown to minimize prediction errors, indicating that the neural network learns an optimal dynamic model of an external process.

This work was done by Tuan Duong, Vu Duong, and Roland Suri of Caltech for NASA's Jet Propulsion Laboratory.

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:

Innovative Technology Assets Management
JPL
Mail Stop 202-233
4800 Oak Grove Drive
Pasadena, CA 91109-8099
(818) 354-2240
E-mail: iaoffice@jpl.nasa.gov

Refer to NPO-41691, volume and number of this NASA Tech Briefs *issue, and the page number.*

Evolutionary Computing Methods for Spectral Retrieval

Solutions to the inverse problem of spectral retrieval are found in a computationally efficient process.

NASA's Jet Propulsion Laboratory, Pasadena, California

A methodology for processing spectral images to retrieve information on underlying physical, chemical, and/or biological phenomena is based on evolutionary and related computational methods implemented in software. In a typical case, the solution (the information that one seeks to retrieve) consists of parameters of a mathematical model that represents one or more of the phenomena of interest.

The methodology was developed for the initial purpose of retrieving the desired information from spectral image data acquired by remote-sensing instruments aimed at planets (including the Earth). Examples of information desired in such applications include trace gas concentrations, temperature profiles, surface types, day/night fractions, cloud/aerosol fractions, seasons, and viewing angles. The methodology is also potentially useful for retrieving information on chemical and/or biological hazards in terrestrial settings.

In this methodology, one utilizes an iterative process that minimizes a fitness function indicative of the degree of dissimilarity between observed and synthetic spectral and angular data. The evolutionary computing methods that lie at the heart of this process yield a population of solutions (sets of the desired parameters) within an accuracy represented by a fitness-function value specified by the user. The evolutionary computing methods (ECM) used in this methodology are Genetic Algorithms and Simulated Annealing, both of which are well-established optimization techniques and have also been described in previous *NASA Tech Briefs* articles. These are embedded in a conceptual framework, represented in the architecture of the implementing software, that enables automatic retrieval of spectral and angular data and analysis of the retrieved solutions for uniqueness. This framework is composed of three modules (see figure):

1. The central core, which consists of the aforementioned ECM;
2. The synthetic-spectra generator, which, coupled with the ECM, generates a population of automatically retrieved spectral solutions; and
3. The synthetic-spectra degeneracy analyzer ("degeneracy" is used here in the mathematical sense of signifying the existence of multiple equally valid solutions for a given set of data and user-defined accuracy), which applies several well-established mathematical methods to characterize the uniqueness (or the degeneracy, which is essentially the lack of uniqueness) of the solutions within the population.

One advantage afforded by this ECM-based methodology over traditional spectral retrieval methods is the ability to perform an automatic, unbiased search for all solutions within the entire parameter