1995 American Control Conference. June 21-23, 1995. Seattle, Washington

# Reconfigurable Control of a Free-Flying Space Robot Using Neural Networks

Edward Wilson[1] Stephen M. Rock[2]
Stanford University
Aerospace Robotics Laboratory
Stanford, California 94305
ed,rock@sun-valley.stanford.edu

**Abstract**

*An experimental demonstration of a new, reconfigurable neural-network-based adaptive control system is presented. The system under control is a laboratory model of a free-flying space robot whose position and attitude are controlled using eight on-off air thrusters. The neural-network controller adapts in real-time to account for multiple destabilizing thruster failures.*

*The adaptive control system is similar in structure to a conventional indirect adaptive control system. While a system identification process builds a model of the robot, a neural controller is trained concurrently using backpropagation to optimize performance using this model. The active controller is updated every few seconds, yielding quick adaptation. Stability is restored within 4 seconds, system identification is complete within 48 seconds, and near-optimal performance is achieved within 2 minutes.*

## 1. Introduction

The nonlinear, parallel, and adaptive capabilities of neural networks make them promising for control applications. Neural networks derive their advantage in solving very complex problems from the emergent properties that come with the massive interconnection of simple processing units. With good training techniques, the networks are capable of implementing very complex behaviors.

Numerous examples in the literature demonstrate the potential of neural-network control [1]. There are three important issues which often arise in a real-world control application that have not been effectively addressed in the NN literature, however:

1. *A priori* knowledge is often available in the form of models of the system's key components and a preliminary control design (e.g., provided by "conventional" control design techniques). Is it possible to use this *a priori* information to improve greatly the performance the neural network can then enable?
2. Many control applications involve the use of discrete-valued devices. For example, thrusters often operate "on-off" rather than with analog-valued outputs. This presents a problem for backpropagation learning, since these discrete-valued functions are not continuously differentiable. Is it possible to modify backpropagation to accommodate the discrete-valued functions?
3. Speed of learning is often important in real-time control applications. Can backpropagation learning be made fast enough to be feasible for rapid on-line adaptation?

In addition to the development of a reconfigurable thruster control system, the goal of the work reported here was to develop extensions to neural-network theory that would address each of these issues. These developments are reported in the context of the robot control application that follows.

## 2. Robot Control Application

The control task addressed in this research is the control of position and attitude of a free-flying space robot using on-off thrusters. Control using on-off thrusters is an important problem for real spacecraft [2], and the nonlinear and adaptive capabilities of neural networks make them attractive for this application. A NN-based approximation method scales well to higher-dimensional thruster controllers, as well as providing a structure conducive to reconfigurable control. Further details regarding the robot control application are presented in [3] [4].

The challenge presented here is to damage mechanically a number of thrusters (as in Figure 2), and then have the control system autonomously and rapidly reconfigure itself in real time. Some thruster failures are strongly destabilizing, which places high demands on the speed of recovery.

[1] Ph.D. Candidate, Department of Mechanical Engineering. Research supported by NASA and AFOSR.

[2] Associate Professor, Department of Aeronautics and Astronautics.

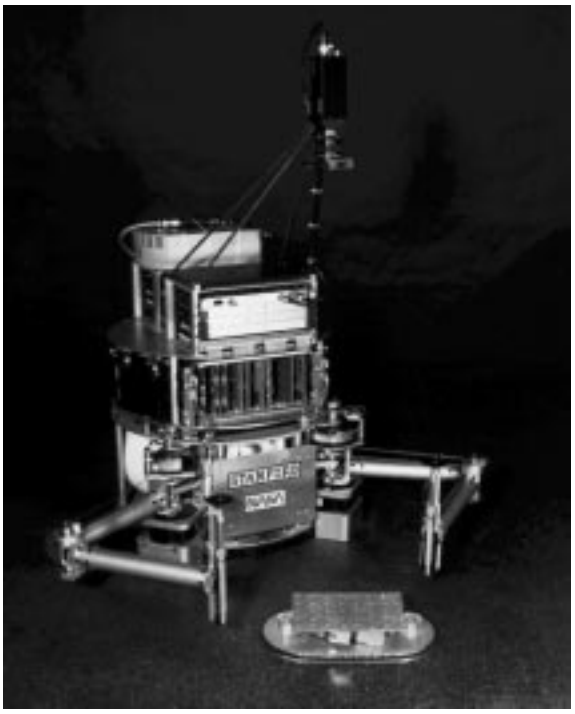

**Figure 1: Stanford Free-Flying Space Robot**

*This highly-autonomous mobile robot operates in the horizontal plane, using an air-cushion suspension to simulate the drag-free and zero-g characteristics of space.*

The experimental equipment, shown in Figure 1, is a fully-self-contained planar laboratory-prototype of an autonomous free-flying space robot complete with on-board gas, thrusters, electrical power, multiprocessor computer system, camera, wireless Ethernet data/communications link, and two cooperating manipulators. It exhibits nearly frictionless motion as it floats above a granite surface plate on a 50 micron thick cushion of air [5]. Accelerometers and an angular-rate sensor sense base motions.

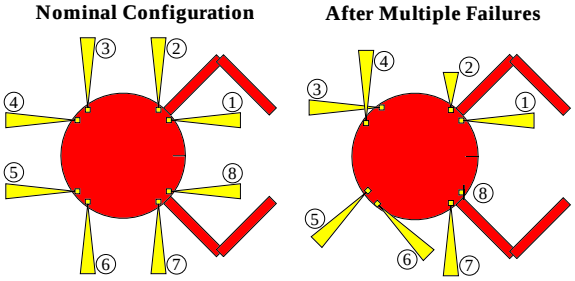


**Figure 2: Example failure mode**

*Magnitude and direction of each of the eight thrusters is shown. Thruster failures were simulated mechanically with weaker thrusters and 45° and 90° elbows. Some elbows destabilize the robot by changing the sign of the thrust in the $\psi$ direction.*

The three degrees of freedom $(x, y, \psi)$ of the base are controlled using eight thrusters positioned around its perimeter, as shown in Figure 2. Each thruster produces both a torque and net force on the robot. This coupling, combined with the on-off nature of the thrusters, substantially complicates the control task.

The robot-base-control strategy developed for this system is shown in Figure 3. The thruster mapping task that must be performed during each sample period is to take an input vector of continuous-valued desired forces and torques, $[F_{xdes}, F_{ydes}, \tau_{\psi des}]$, and find the optimal output vector of discrete-valued (off, on) thruster values, $[T_1, T_2, ..., T_8]$.

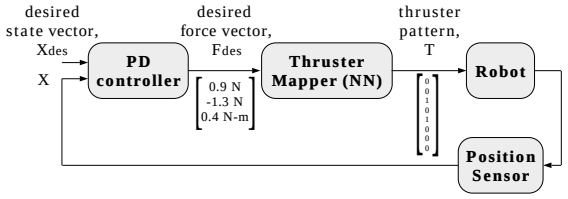


**Figure 3: Robot-Base Control**

*The PD control module treats the thrusters as continuous actuators. The thruster mapper must find the thruster pattern that will produce a force closest to that requested by the base-control module.*

In [5], a conventional approach is presented that solves this problem by performing an exhaustive search at every sample period. With eight bi-level thrusters, 256 possible thruster combinations exist. By clever use of symmetries, this search space was reduced to 11, making the problem tractable. Unfortunately, this solution method does not scale well for a three-dimensional robot, or when thruster failures are allowed, disrupting the symmetries.

This provides the motivation for using a neural network: *the neural network is used to implement a nonlinear approximation to the optimal solution – one that can be computed in real-time.*

## 3. Reconfigurable Control System

Often the most important, and sometimes the most difficult aspects of a neural-network control application are the decisions about how to structure the control system and which components are to be neural-network-based. The first issue is to determine whether the application is one where neural networks can contribute efficiently better (and cheaper) control than is achievable without them. If they can, the second issue is to determine in just which segment(s) of the control system they should be used in order to do so.

To determine where neural networks can contribute effectively, the control systems engineer must consider the strengths of neural networks (nonlinear, adaptive, generic, unstructured, parallelizable) as well as the costs associated with these benefits (difficult to understand workings or prove stability, design is iterative, computationally complex). The cost/benefit balance must be evaluated on an application by application basis. First at the system level, the system requirements and considerations of degree-of-nonlinearity, adaptation requirements, and computational complexity, etc., lead to a candidate system architecture. Then at the component level, this cost/benefit analysis is repeated, leading to the decision of what sort of subsystem will be used in each segment of the control system [4].

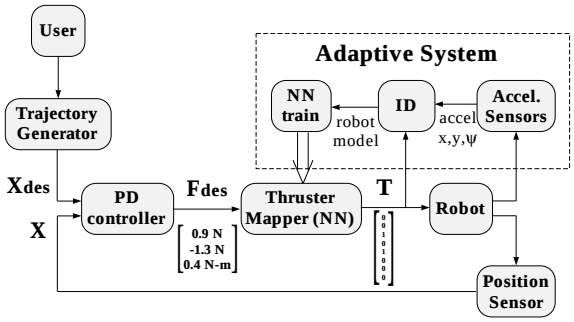


**Figure 4: Reconfigurable Control System**

*This control system is based upon a conventional indirect adaptive controller, such as a self-tuning regulator. The ID block represents a recursive least squares identification of thruster strength and direction. This continually-updated model is passed to the NN training block, shown in detail in Figure 5. The continually-updated neural thruster mapper is loaded into the active control loop every few seconds.*

Applying these principles to the robot control application has resulted in a structure, shown in Figure 4, that is modelled after a standard control architecture known as "indirect adaptive control." "Indirect" refers to the use of sensor information to build a model of the system, and then to redesign a controller based upon the updated plant model.

A recursive linear regression component was used for failure detection and identification, since identification of the thruster characteristics is a linear process. The algorithm used to obtain acceleration measurements was nonlinear, but could be derived analytically, so no neural network was used there either.

A neural network was used precisely at the location where it is beneficial: the thruster mapper. This is an inscrutable nonlinear function that requires adaptation, and the benefits of a neural-network approach do indeed outweigh the costs. The control redesign process is therefore a backpropagation-based neural-network training algorithm.

## 4. Neural Network Developments

In development of the control system, two major neural network challenges were faced that resulted in development of a new NN architecture and training method. These developments are mentioned briefly here, and in more detail in [3] [4] [6].

### 4.1. Fully-Connected Architecture

To speed up the reconfiguration process, a general neural-network architecture was developed. This "Fully-Connected Architecture" is for feedforward neural networks that can be trained using backpropagation [7].

The FCA has many advantages over a layered architecture [3] [6]. For this application, the most significant advantage comes from the feedthrough weights. These weights provide a direct, linear connection matrix from inputs to outputs (provided sigmoids are used only on hidden units). This produces fast initial learning, and allows *direct pre-programming* of a linear solution calculated by some other method. This is especially important for control applications, where a large body of linear control knowledge exists that can be drawn upon to provide a good starting point. The FCA provides a seamless integration of linear and nonlinear components.

In this application, a linear approximate solution is calculated very quickly based on a pseudo-inverse of a linearized plant model. Injecting this approximate linear solution into the network immediately after a failure is detected results in rapid stabilization of the robot.

### 4.2. Backpropagation Learning for Discrete-Valued Functions

Training a neural network to produce a thruster mapping based upon a model of the robot can be thought of as learning the inverse model of the robot-thruster system, as in Figure 5. This is a common approach, and would be relatively straightforward if not for the discrete-valued functions that represent the on-off thrusters. Some modification to the learning algorithm is required to allow gradient-based optimization to be used with these non-differentiable functions.

The method, shown in Figure 5, relies on approximation of the discrete-valued functions with "noisy sigmoids" during training. This is a broadly-applicable

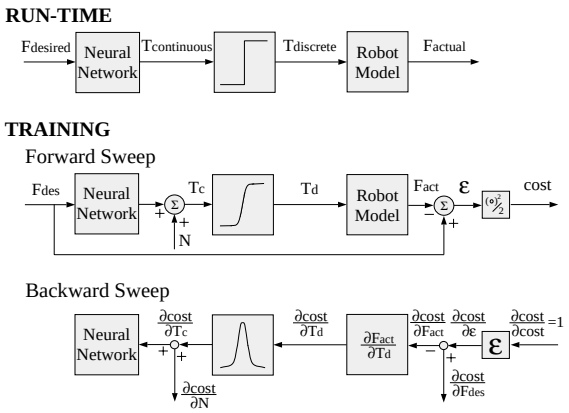


**Figure 5: Thruster mapping, on-line training method**

*This is the training structure used to adapt the thruster mapper during reconfiguration. This process appears as the "NN train" block in Figure 4. The "Robot Model" contains the magnitude and direction of each thruster. During adaptation, this model is updated continually by the "ID" process shown in Figure 4.*

algorithm that applies to any gradient-based optimization involving discrete-valued functions. It is described in detail in [4] [8].

## 5. Experimental Demonstration

Position and attitude of the robot base are controlled while subject to multiple, large, possibly-destabilizing changes in thruster characteristics. An off-board vision system provides high-bandwidth position feedback, which is then digitally filtered and differentiated to provide velocity feedback. On-board accelerometers and an angular-rate sensor are used to provide base-acceleration measurements used by the failure-detection and control-reconfiguration capability.

The FCA neural-network thruster-mapping component described in Section 3 was implemented on the on-board Motorola 68040 processor, as was the rest of the control system (at a 10 Hz sample rate).

Figure 2 shows the robot thruster layout in nominal and failed configurations. The magnitude and direction of each thruster is shown. Nominally, each thruster produces 1 Newton of force, directed as shown. The failures were produced by mechanically changing the thrusters. Failures include: half-strength, plugged completely, angled at 45°, and angled at 90°. The 90° failure mode places high demands on the control reconfiguration system, since it destabilizes the robot (changing the direction of torque results in positive feedback).

### 5.1. Failure Detection

Accelerometers and an angular-rate sensor (that is digitally differentiated) produce acceleration signals in $(x, y, \psi)$. These signals are filtered and passed to a system identification process based upon recursive least squares. The parameters identified are the accelerations in $(x, y, \psi)$ resulting from each thruster firing. When a failure is detected, the thruster is excited artificially to speed up the identification process. For the case presented here, with 6 of 8 thrusters failed, the ID process took about 48 seconds from when the first thruster fired until the last thruster was identified to a high level of confidence. Since the neural network is training in parallel with this process, stabilization occurs within seconds, and the robot remains well-controlled during the identification.

Results from the reconfiguration are shown in Figure 6. The static control deadband is approximately $\pm$ 3 cm in translation and $\pm$ 11° in rotation. Due to noisy sensors and multiple thrusters firing simultaneously, it takes about 4 seconds before the ID process is confident enough to confirm a thruster failure and begin reconfiguration. An initial stabilizing reconfiguration occurs nearly instantaneously as a linear approximate solution is quickly calculated and implemented via the Fully Connected Architecture. Once failures have been detected, they are labelled as "suspects" and are excited artificially to expedite identification.

The neural-network training process shown in Figure 5 runs in parallel with the ID process, with the robot model updated continually by the ID. The network being trained is copied periodically into the network that is controlling the robot.

### 5.2. Rapid Reconfiguration

Very rapid learning is possible due first to the FCA, and due second to the growing of the network. With few hidden neurons, quick learning takes place since fewer computations are required, and fewer training patterns are required (to avoid overfitting). The network begins with 3 inputs, 8 hidden neurons, and 8 outputs, and gradually grows to 30 or more hidden neurons as training progresses. New hidden neurons are added when performance begins to plateau. To prevent overfitting, the training-set size is grown proportionally with the number of hidden neurons. With this arrangement, a mapping with about 30% error above optimal results in 30 seconds, 20% above optimal within 60 seconds, and 10% above optimal[1]

[1] Due to the use of discrete-valued actuators, there is almost always a force error vector. The error value reported here indicates that the average magnitude of the force error vector is 1.10 times the magnitude achievable with an exhaustive search.

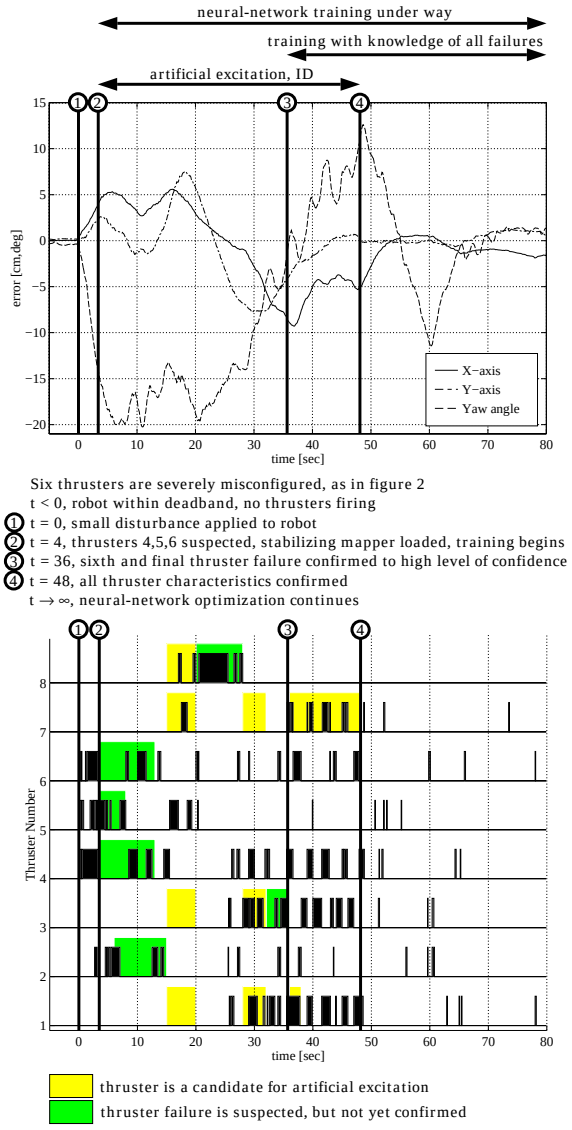


**Figure 6: Reconfiguration Experiment, Position Errors, Thruster Firing**

*$[x,\ y,\ \psi]$ position errors and thruster signals are plotted during autonomous reconfiguration. Black rectangular regions indicate periods of thruster firing. Darkly-shaded regions indicate the time during which the thruster was suspected. In addition to artificial excitation of the suspected thrusters, excitation of other thrusters is used to expedite the identification process. These periods are indicated by the lightly-shaded regions. The robot begins at rest within the deadband, is disturbed at $t = 0$, stabilizes itself within 4 seconds, and completes identification (aided with artificial excitation) after 48 seconds. The neural-network thruster mapper continues to optimize after the identification is complete.*

within 300 seconds. As more hidden neurons are added, the network performance approaches optimality, but at the expense of a slower training rate.

## 6. Summary and Conclusions

This paper has presented an adaptive neural-network-based control system for a free-flying space robot. This control system has a structure that is modelled after a conventional indirect adaptive controller, with the neural network used to implement the nonlinear adaptive component. Specific procedures for determining neural-network applicability are outlined in the paper.

Two neural-network-control developments were critical to achieving quick adaptation and a near-optimal controller. First, a "Fully-Connected Architecture" (FCA) was used that has the ability to incorporate an *a priori* approximate linear solution instantly; this permits quick stabilization by an approximate linear controller. Second, a learning method was used that allows gradient-based optimization (backpropagation) with discrete-valued functions, in this case, the on-off thrusters.

The control system was demonstrated experimentally on a laboratory prototype robot, where stable reconfiguration to major destabilizing thruster failures occurred within 4 seconds, and near-optimal control was achieved within minutes.

**References**

[1] W. Thomas Miller III, Richard S. Sutton, and Paul J. Werbos, editors. *Neural Networks for Control.* Neural Network Modeling and Connectionism. The MIT Press, Cambridge, MA 02142, 1990.

[2] James R. Wertz, editor. *Spacecraft Attitude Determination and Control.* Kluwer Academic Publishers, Boston, 1978.

[3] Edward Wilson and Stephen M. Rock. Neural network control of a free-flying space robot. *Simulation*, June 1995.

[4] Edward Wilson. *Experiments in Neural Network Control of a Free-Flying Space Robot.* PhD thesis, Stanford University, Stanford, CA 94305, March 1995.

[5] Marc A. Ullman. *Experiments in Autonomous Navigation and Control of Multi-Manipulator, Free-Flying Space Robots.* PhD thesis, Stanford University, Stanford, CA 94305, March 1993.

[6] Edward Wilson and Stephen M. Rock. Experiments in control of a free-flying space robot using fully-connected neural networks. In *Proceedings of the World Congress on Neural Networks*, volume 3, pages 157–162, Portland OR, July 1993.

[7] Paul J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, Cambridge, MA 02142, August 1974.

[8] Edward Wilson. Backpropagation learning for systems with discrete-valued functions. In *Proceedings of the World Congress on Neural Networks*, volume 3, pages 332–339, San Diego CA, June 1994. International Neural Network Society.