SANDIA REPORT

SAND97–0957 • UC–706
Unlimited Release
Printed April 1997

Artificial Awareness for Robots

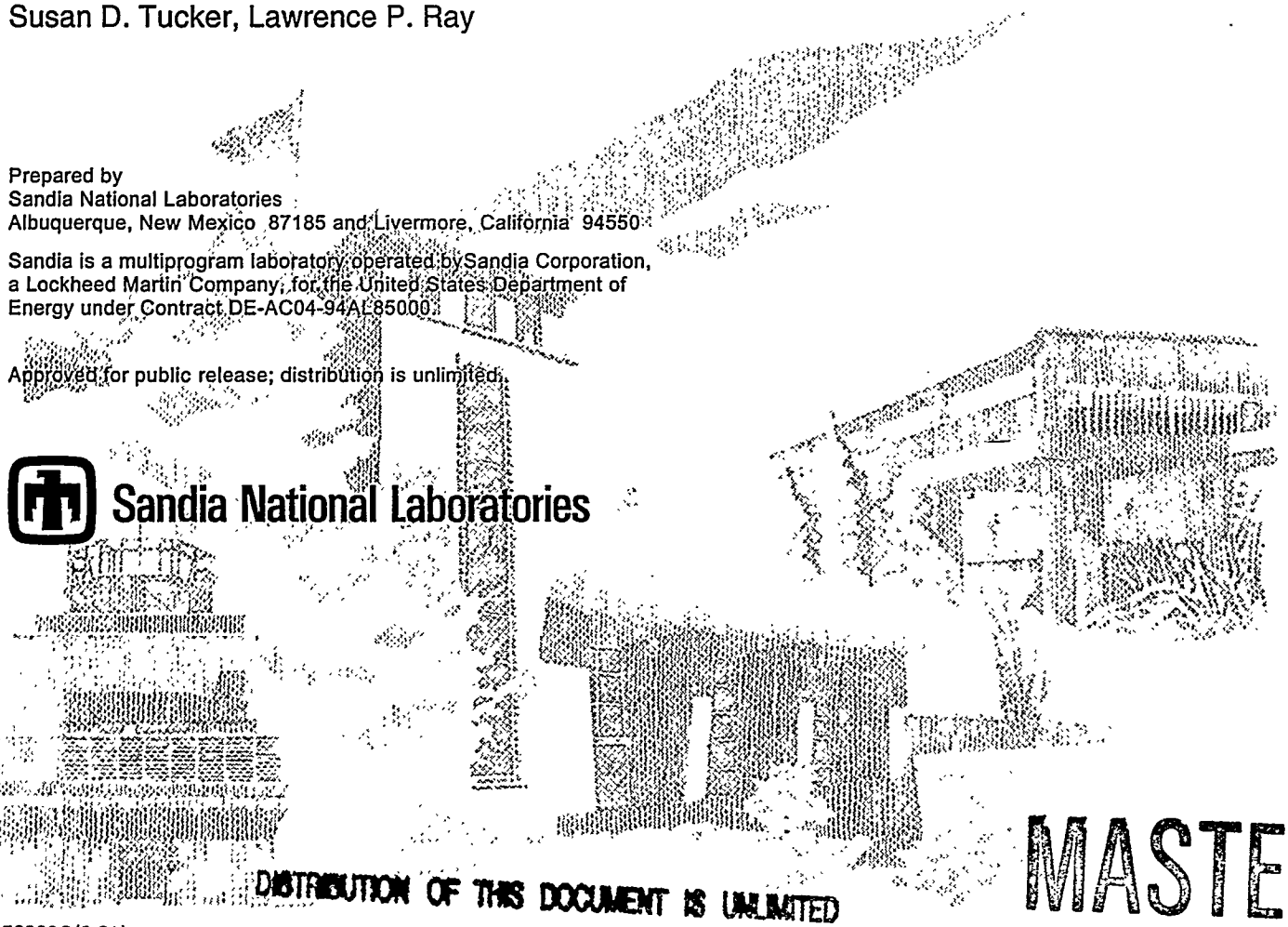
Using Artificial Neural Nets to Monitor Robotic Workcells

Susan D. Tucker, Lawrence P. Ray

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Approved for public release; distribution is unlimited.



Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A04
Microfiche copy: A01

Intentionally Left Blank

Contents

Intentionally Left Blank

Artificial Awareness for Robots

Using Artificial Neural Nets to Monitor Robotic Workcells

1. Purpose and Experimental Setup

1.1 Purpose

The purpose of this work is to investigate the feasibility of a system which detects unusual operating conditions in a robotic workcell and warns the operator or suspends operations. This objective was inspired by the observation that humans become accustomed to the normal pattern of operations in a robot workcell and are very sensitive to changes in that pattern. Typically, people working near a workcell, but not operating it, completely ignore routine operations and may even be unable to say whether the robot has been operating. However, when an operation deviates from the norm, everyone is alerted and looks up to see what the problem is.

In contrast, current robotic systems are unable to recognize most unexpected changes in the work environment, such as tool breakage, workpiece motion, or sensor failure. Unless halted by a human operator, they are likely to continue actions that are at best inappropriate, and at worst may cause damage to the workpiece, tooling, or robot. This project developed a sensor analysis system which learns the expected characteristics of sensor data during normal operations, recognizes when data are no longer consistent with normal operation, and can suspend operations and alert a human operator. Artificial Neural Networks were trained to monitor the data stream and signal conditions not consistent with previous iterations of the same task.

Humans seem to rely heavily on what they hear to detect abnormal workcell operation. For the automated system, though, we chose to depend on data from force/torque sensors, which measure force along and torque about three orthogonal axes. The sensor is able to compute the force and torque under the assumption that they are applied at an arbitrary origin; typically a nominal point about which the tool acts (the "tool center point") is designated as the origin. The axes of measurement may also be rotated to align with tool features. Force/torque sensors are frequently incorporated in robotic workcells for control purposes, and give information closely

related to task performance. Using this data instead of audio data avoids the problem of ignoring unusual sounds which are unrelated to the workcell operation.

Data on force and torque applied at the robot tool center point were collected from two workcells: a robotic deburring system and a robot material-handling system. Data were collected both for normal operations and for operations in which a fault condition was introduced. Data simulating excessive sensor noise and sensor failure were generated by appropriate alteration of collected data under normal conditions. These data were used to characterize failure conditions that could not readily be created in the workcells. Artificial Neural Networks (ANN) were trained to classify these operating conditions; several ANN architectures were tested.

1.2 Material Handling Workcell

Force and torque data for a material handling task were collected from the Component HAzardous Material Precision Separation System (CHAMPS). Figure 1 shows the primary equipment in this workcell. The process is broadly representative of material handling; the robotic operations consisted of the following tasks:

1. Get the gripper by attaching it using the EOA tool adapter. This is referred to as the GETGRIP operation.
2. Get the workpiece by gripping it at the input station (GETWORK operation).
3. Put the gripper, still holding the workpiece, in the X-ray station (PUTXRAY operation).
4. Get the gripper and workpiece from the X-ray station (GETXRAY operation).
5. Put the gripper, still holding the workpiece, in the cutting station (PUTCUT operation).
6. Get the gripper and workpiece from the cutting station (GETCUT operation).
7. Put the workpiece in the output station (PUTWORK operation).
8. Put away the gripper (PUTGRIP operation).

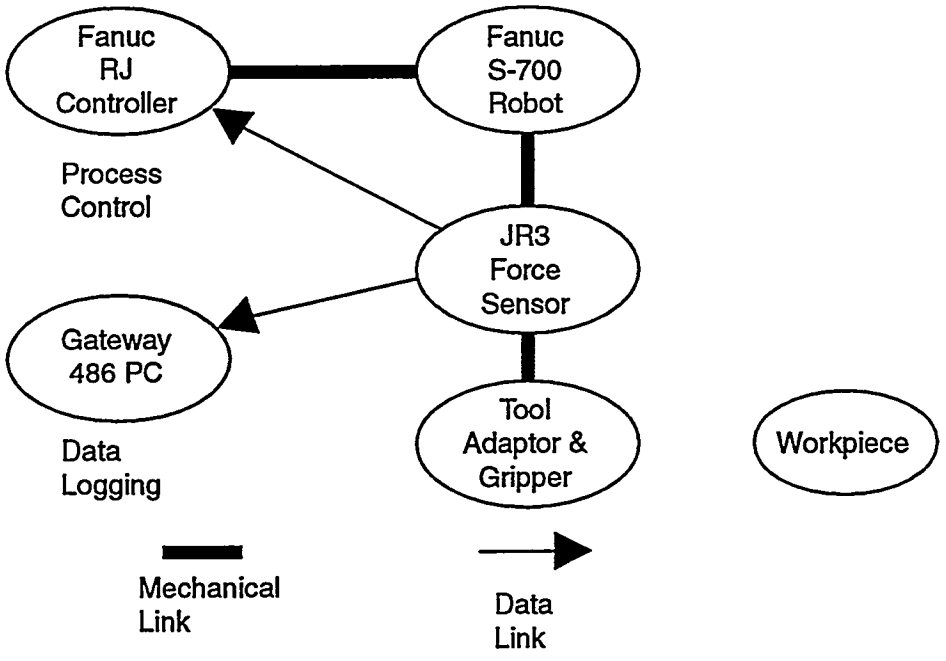


Figure 1- Material Handling Workcell

The JR3 force sensor was configured to output force and torque data to both an RS-232 link for process control and a high-speed parallel communication link for data logging. The data were logged by a Gateway 2000 4DX2-66 computer. Forces along three orthogonal axes and torques about those axes were recorded. The force sensor was programmed to compute the forces and torques assuming they were applied at the tool center point (TCP) of the gripper, along axes aligned with the principal structure of the gripper. Force and torque were sampled at a 400 Hz rate, and were subsampled to a 100 Hz rate for some of the ANN processing.

1.3 Deburring Workcell

Data were also collected from a second workcell which performs a material processing task. This workcell removes burrs from machined metal components, or "deburrs" them. Figure 1 shows the primary equipment in this workcell. For the deburring operation, two tasks were monitored:

1. Make contact with the workpiece edge, or "find" the edge;
2. Move a power-driven wire brush along the edge, or "brush" the edge.

The force sensor was part of optional equipment purchased from Adept and integrated into the Adept robot controller. The data were logged by the controller. The force sensor was programmed to compute forces and torques at the tip of the wire brush, with the Z-axis aligned along the brush shaft. Data were sampled at either a 50 Hz or 500 Hz rate.

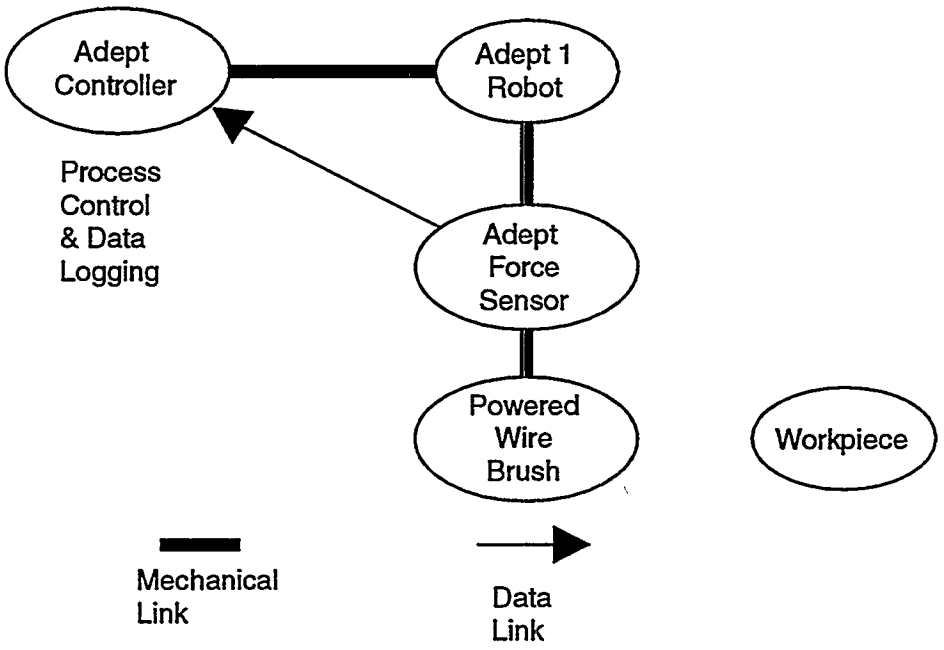


Figure 2 - Deburring Workcell

2. Results from Material Handling Workcell

2.1 Problem description

The purpose of the error detection neural networks is to signal an error condition as soon as possible after it occurs so that the process can be shut down before damage to the robot, workpiece, or environment can occur. The goal in designing a network or set of networks for this application was to develop a simple technique that can be automated. Clearly a neural network failure detection approach will not be a success in an "agile" environment if extensive design effort is required for each operation required of a robot. Additionally, we chose not to add additional sensors in order to collect data that might indicate failure: the goal was to detect problems using data that would normally be available in controlling the movements of the robot.

The material handling robot application used for this study has eight distinct operations to be monitored. Six-axis force / torque data were collected for each of the operations, resulting in 6 channels of data plus time information for each operation. Data rates of 100 Hz and 400 Hz were used. Examination of the 400 Hz data showed that a 100 Hz rate adequately characterized the features in the data, although the Nyquist criterion was not satisfied for high-frequency transients. Examples of this data for representative "good" operations are shown in Figure 3 through Figure 10.

Training of neural networks is accomplished using example data. Fourteen example "good" runs for each operation were collected, as well as data sets in which data was known to be erroneous. Specifically, error set data were taken in which the robot gripper and/or workpiece were missing. Inadvertently, some data were collected in which the measurement axes of the force sensor were incorrectly configured, resulting in additional "bad" data to use as test cases.

This application represents a special problem in that it is not practical to collect data for all of the "bad" data conditions that might occur. While this is not an unusual situation, it presents difficulties to the neural network designer. This is handled in this work by surrounding the "good" data space with "bad" data space, as will be described in Section 2.4.

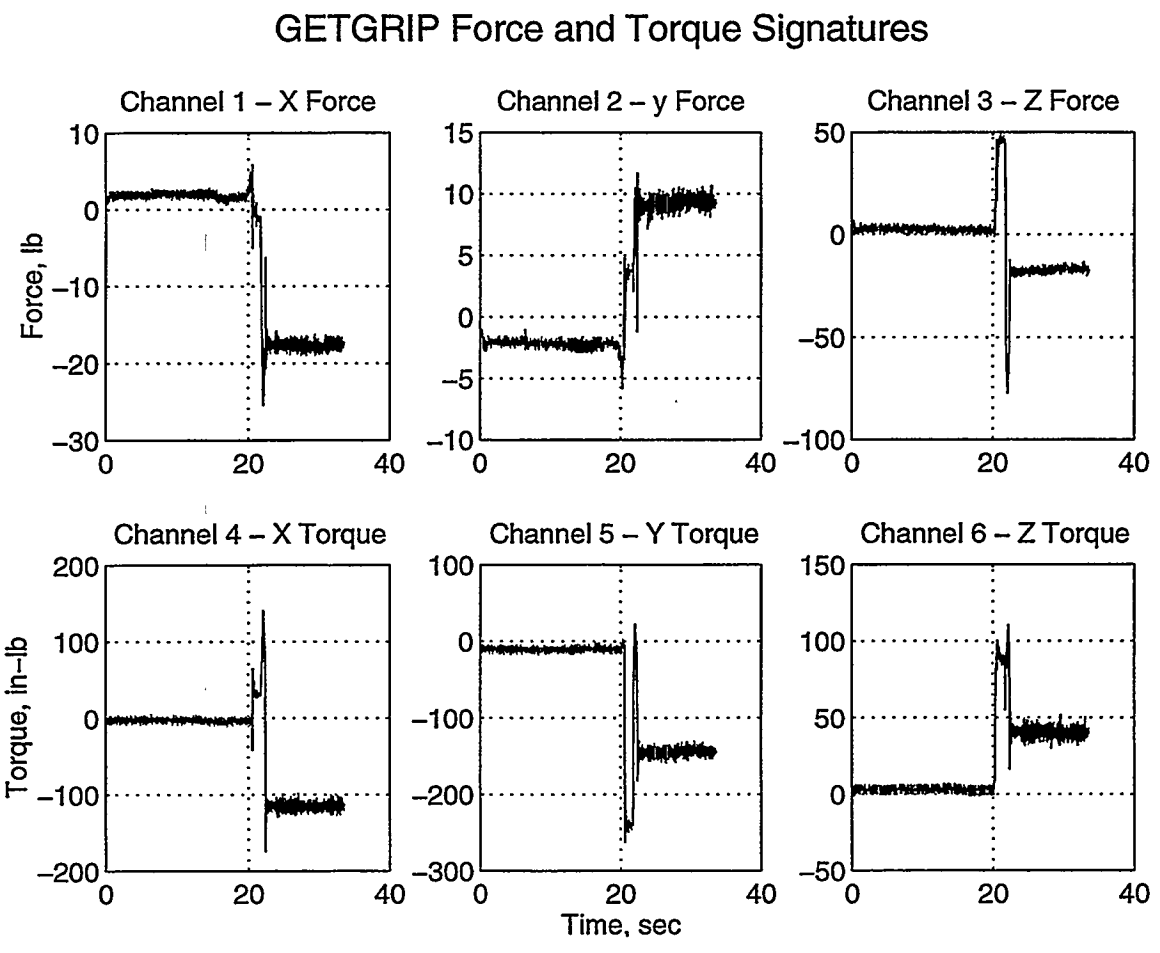


Figure 3 - GETGRIP Force and Torque Signatures

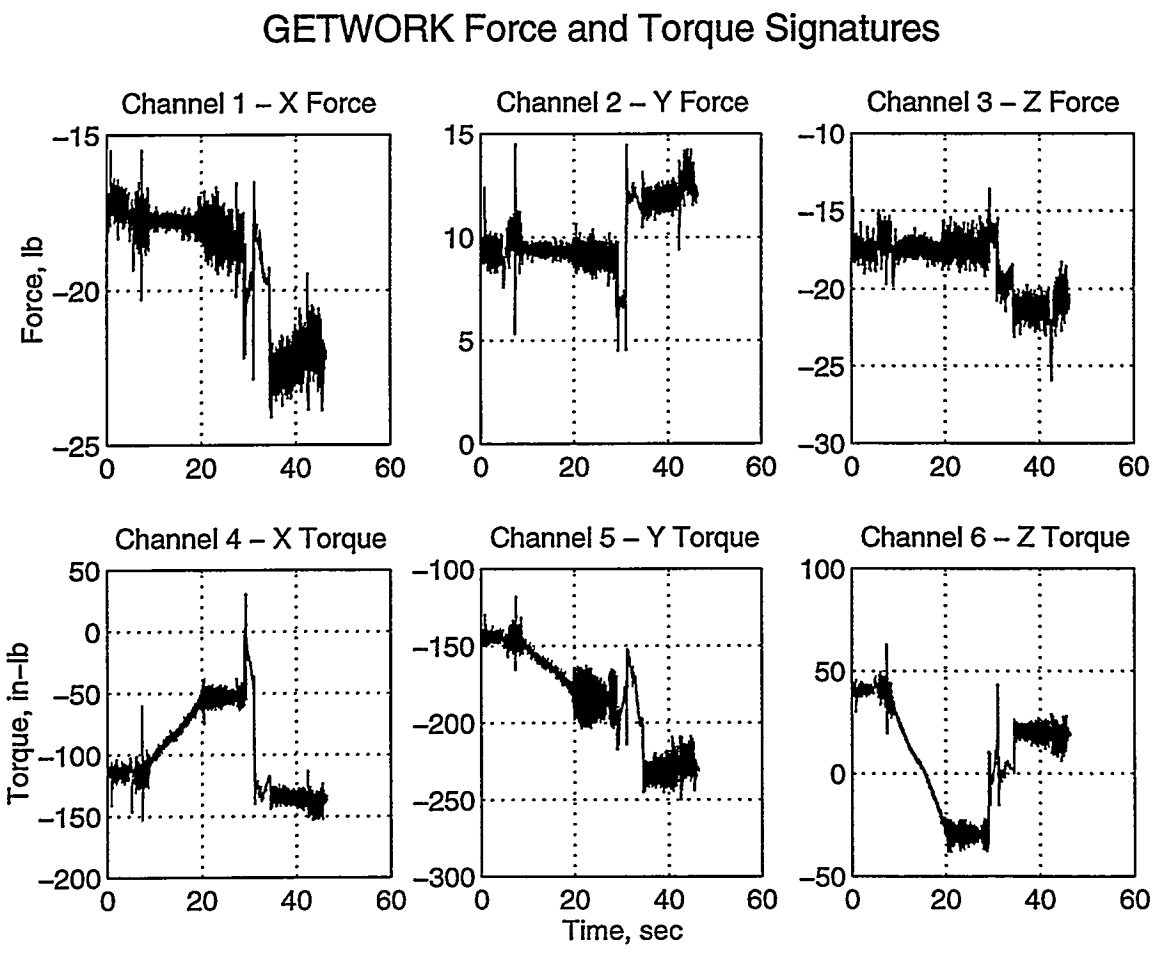


Figure 4 - GETWORK Force and Torque Signatures

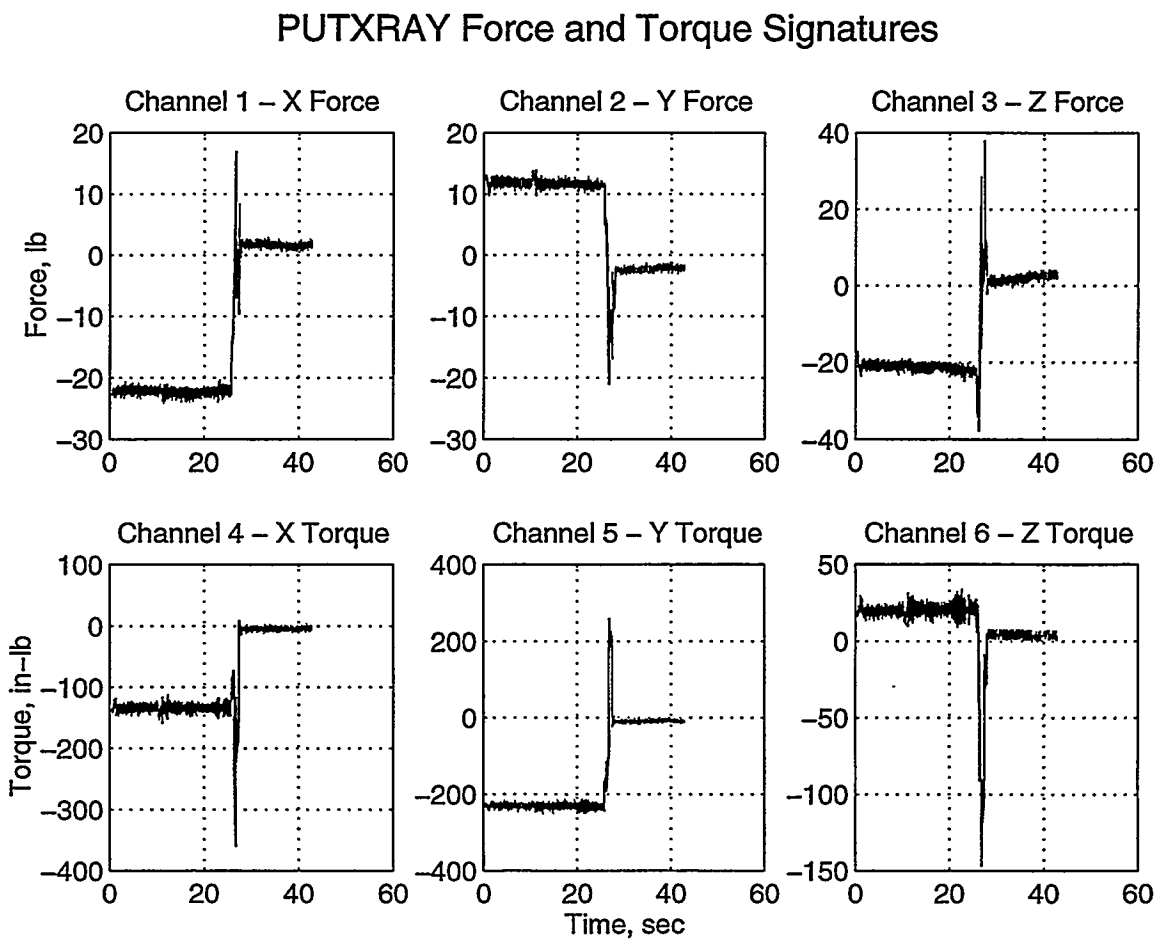


Figure 5 - PUTXRAY Force and Torque Signatures

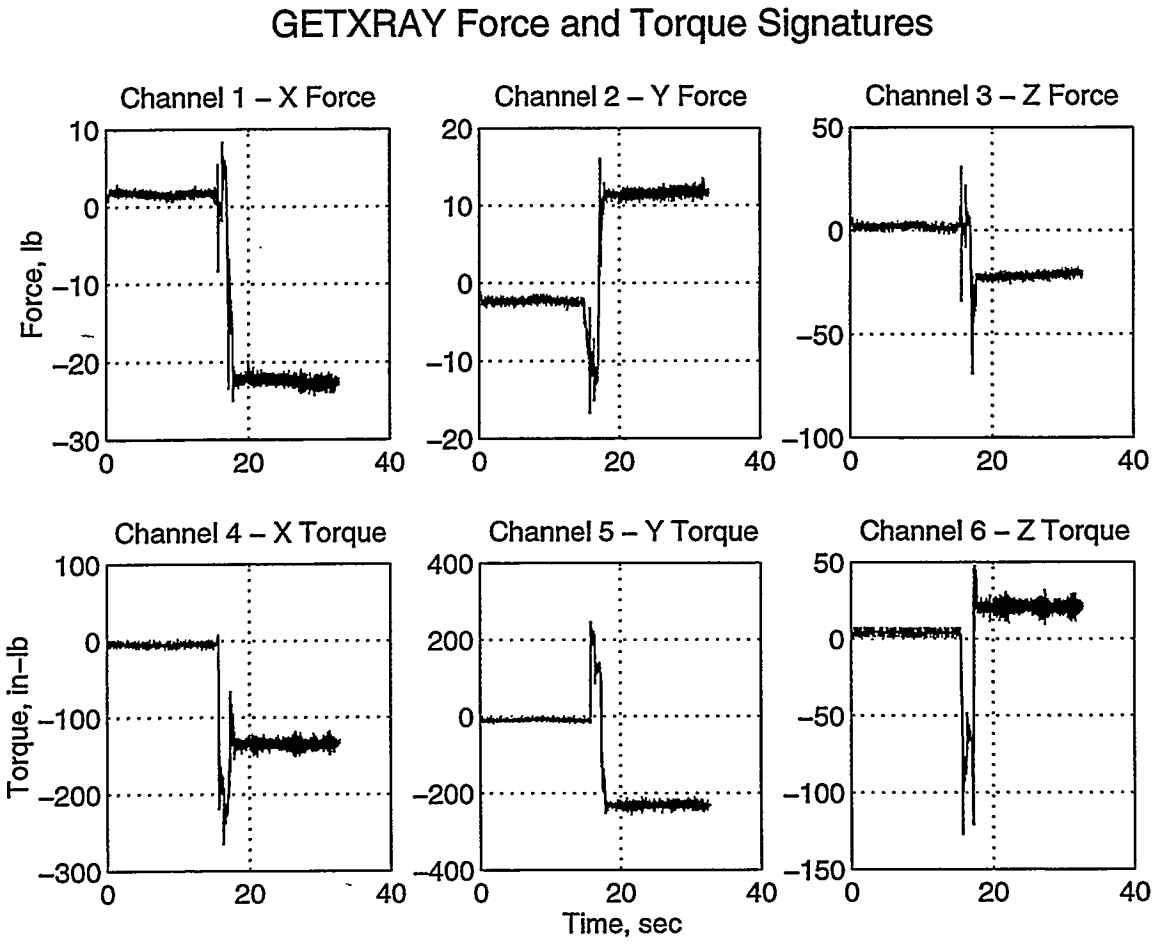


Figure 6 - GETXRAY Force and Torque Signatures

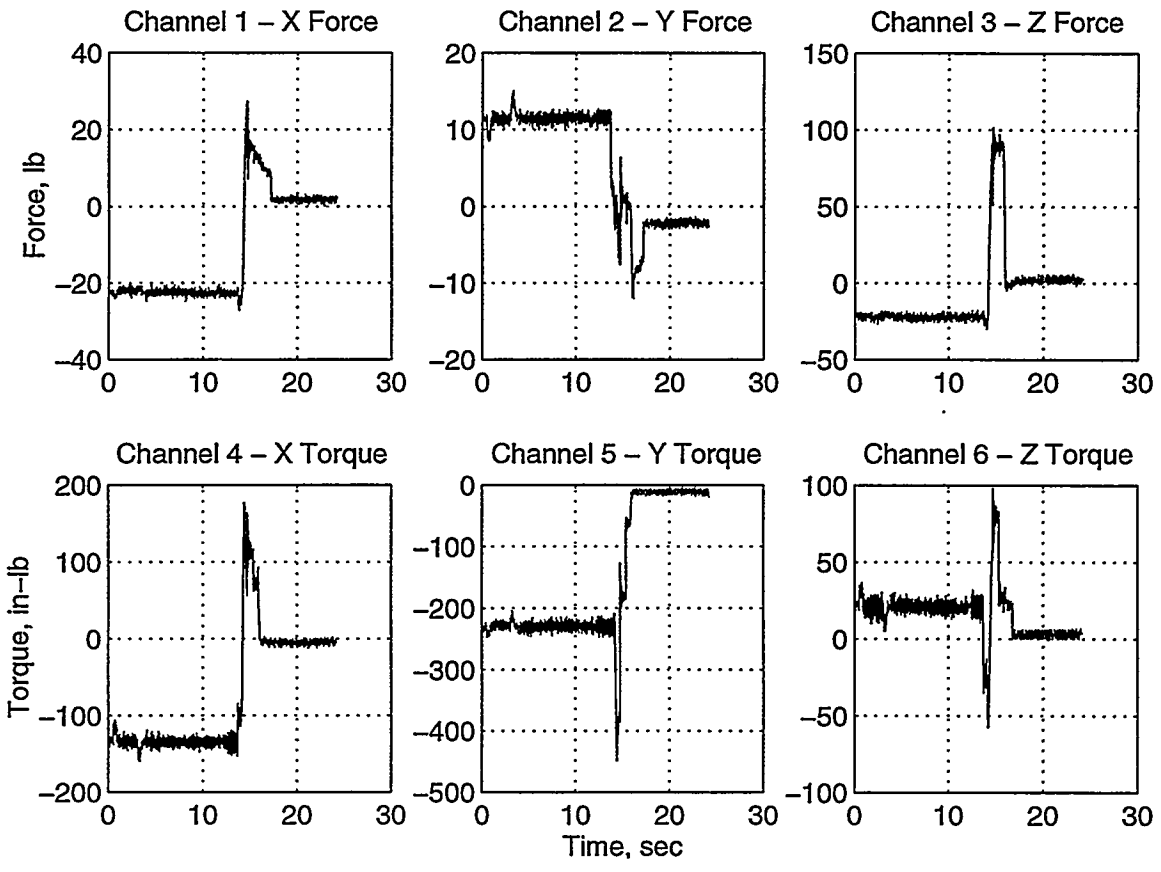


Figure 7 - PUTCUT Force and Torque Signatures

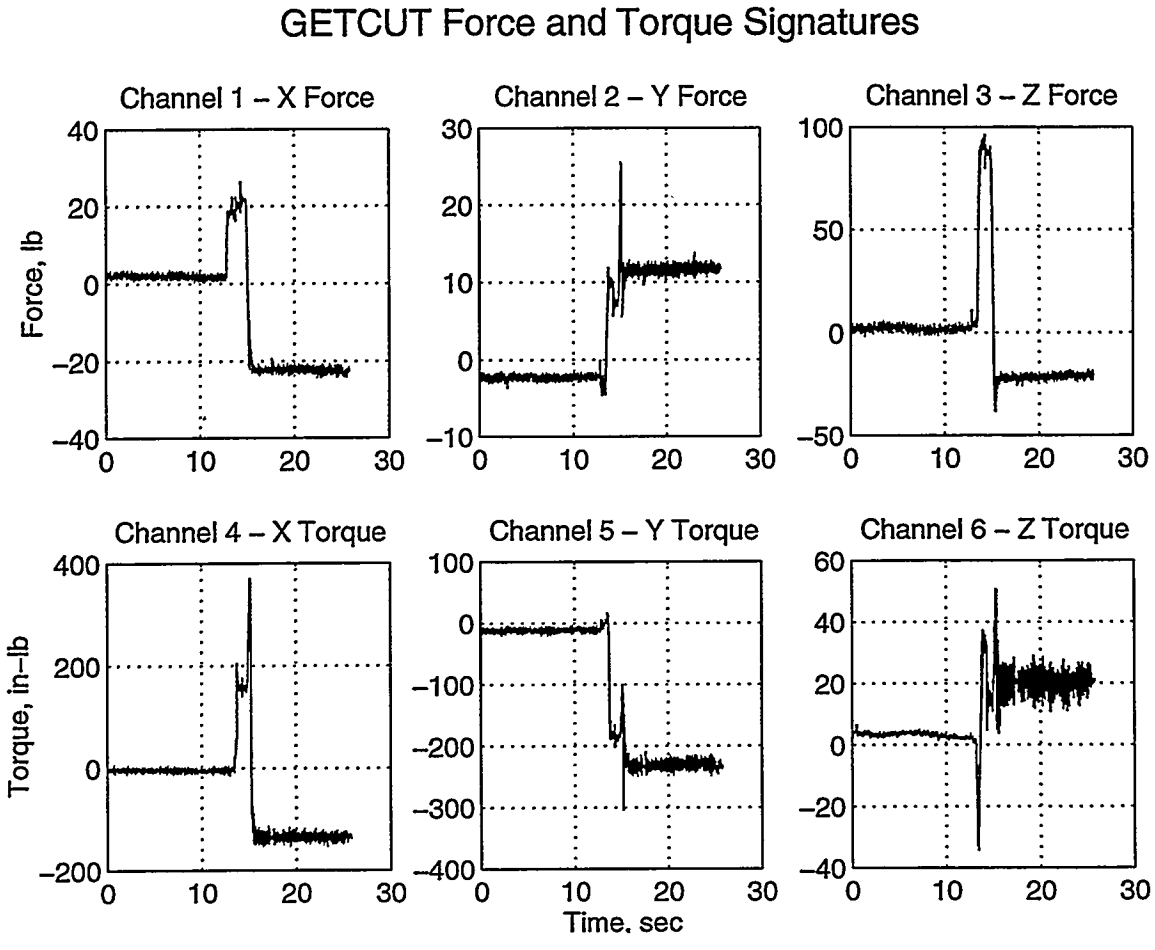


Figure 8 - GETCUT Force and Torque Signatures

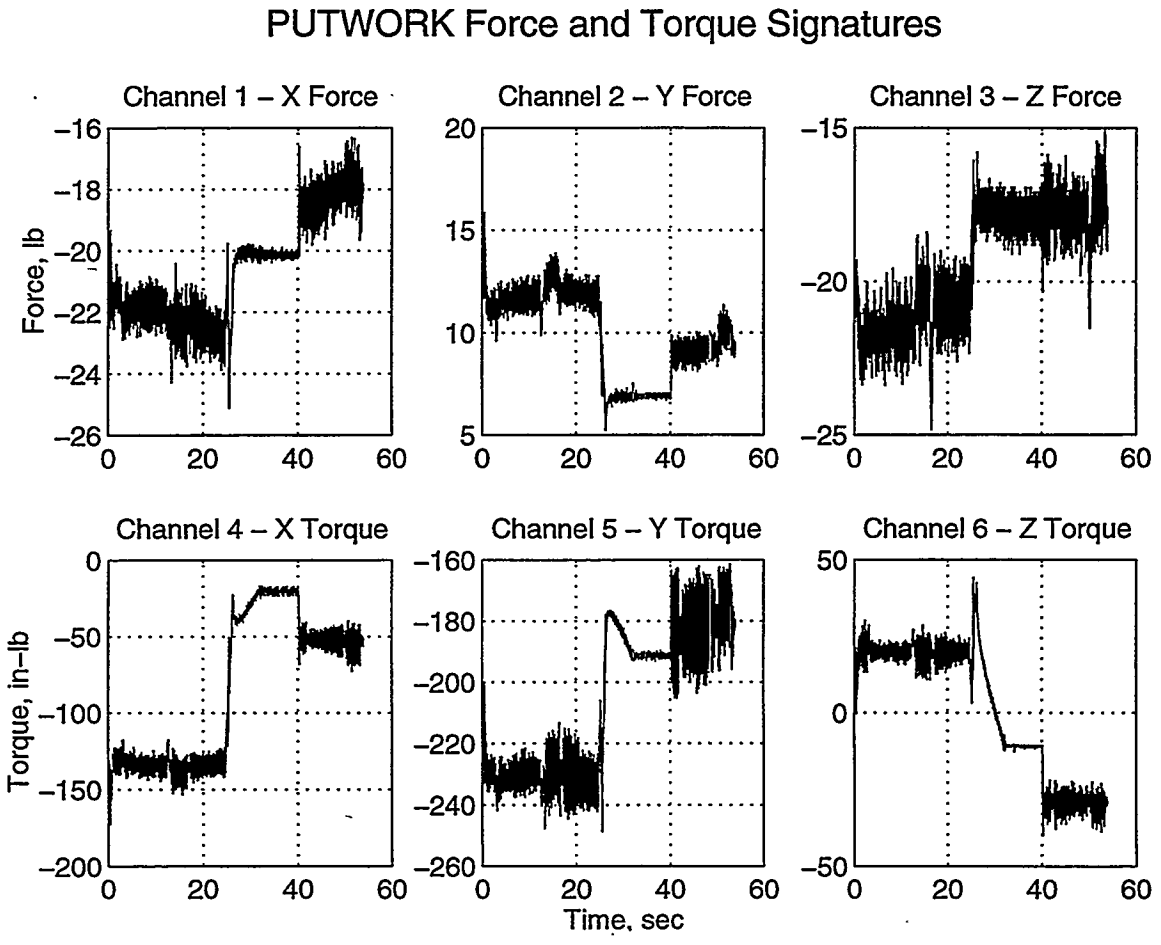


Figure 9 - PUTWORK Force and Torque Signatures

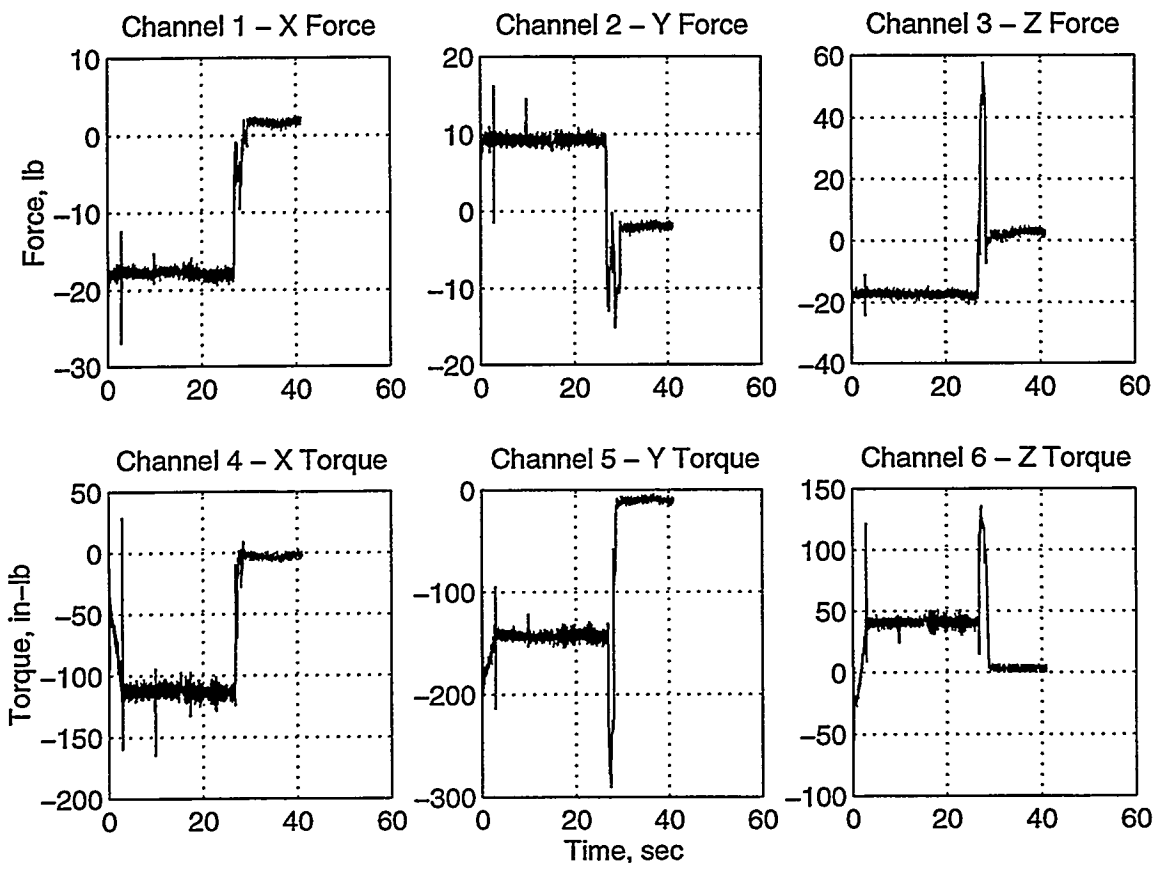


Figure 10 - PUTGRIP Force and Torque Signatures

The most direct approach seemed to be to design a neural network that would detect out of range conditions for each of the 6 channels. Then data from the individual channels is combined to indicate an error whenever one of the 6 channels is in error. Finally, to achieve robustness in the presence of noise or other data problems, an operation is not shut down unless a threshold error rate is reached.

2.2 Neural Network Architecture

The three layer feed-forward neural network described by Chen, et. al., is used for this paper [Chen94]. It is unfortunate that the authors do not assign a name to an architecture they deem to be novel. For want of a better name, it will be referred to as the Chen/Thomas/Nixon architecture in this paper. This type of network is a classification network with the desirable property that 100% correct classification on the training patterns is always attainable. Depending on its operating parameters, the network can behave as a scale-invariant classifier, a nearest-neighbor classifier, or somewhere in-between. For this work, parameters were chosen so that the network behaves as a nearest neighbor classifier. The cost of the 100% correct classification is exacted in memory usage and execution time. This type of neural network will require more memory and computation time to do a task than would a comparable backpropagation network. By comparison, a backpropagation network suffers from the disadvantage of an open-ended amount of training time and experimentation with the network architecture. Another advantage of the chosen architecture is that the training set can be augmented with new data with the assurance that previously learned knowledge will not be disturbed.

It should be mentioned that backpropagation networks were tested on the data for this problem, but the networks did not converge quickly or easily (i.e., without a lot of experimentation). This is not to say that a backpropagation net would not work for this problem, or that the use of some other training algorithm might not make training easier. The empirical nature of backpropagation/feed-forward network design creates a situation in which a satisfactory design might always be possible if the designer just tried "one more thing."

As noted above, the Chen/Thomas/Nixon neural network architecture creates a classification network. Its structure consists of an input layer, two hidden layers, and an output layer. A mathematical formulation of this network is found in [Chen94]. The network may be described as follows:

(1) Input Layer — Input vector is augmented with a constant scalar value R, treated as an additional input, to form the extended input pattern. This scalar value, used also in Hidden Layer 1, is what determines the behavior of the

network as a scale-invariant versus nearest-neighbor classifier.

(2) Hidden Layer 1 — Computes the dot product of the extended input pattern with each normalized, extended, prototype vector. An extended prototype vector is an input vector chosen from the training set during the training phase (see Section 2.3) which is augmented by the same constant scalar input R used to augment the input vector. Each prototype vector is also normalized to have a length of 1.

(3) Hidden Layer 2 — Performs a maximum function on the layer 1 outputs to determine the prototype pattern most similar to the input pattern.

(4) Output Layer — Assigns the class of the winning layer 2 prototype to the input class.

2.3 Training Algorithm

In [Chen94], Chen, et. al describe a training algorithm which initially generates a neural network using each training point as a prototype node. This ensures 100% correct classification on the training data. The learning algorithm then shrinks the network by removing those prototype nodes that do not affect performance on the training set. Unfortunately, there was so much training data for each network in this application that the initial network was extremely large, resulting in intolerable execution times in the early stages of training. As a result, an alternate algorithm to "grow" rather than "shrink" the network was developed.

The network growing algorithm proceeds as follows:

(1) Set up a single node network using the first training point as the prototype.

(2) For each training point (CURRENT_POINT) after the first:

(a) test performance on current point

(b) if classification is correct, do nothing

(c) if classification is incorrect:

1. add CURRENT_POINT to prototype set
2. test performance on points 1 through CURRENT_POINT
3. if classification is correct for all points 1 through CURRENT_POINT, do nothing
4. if classification is incorrect for any points 1 through CURRENT_POINT, add first point with error to prototype set, then resume at step 2.c.2

When this process is completed, the resulting network will have 100% correct classification on the training set.

2.4 Preprocessing and Neural Network Design

As stated above, one network was designed for each data channel (FX, FY, FZ, TX, TY, TZ) and each operation (GETGRIP, etc.). The inputs to the network were time since the beginning of the operation and the data value itself. These were scaled so that the range of time and data both fell in the range 0 to 1.

For each operation and each channel, a network was designed according to the following procedure:

(1) Data were divided into training and test data sets. The training data sets consisted of eleven "good" operations. Test data consisted of two to three "good" operations and two to five known "bad" operations. The "bad" data were evaluated by a human who determined at what times the "bad" operations moved from "good" to "bad" and vice versa.

(2) Create one large data set from the training data files. Inputs to the network are TIME and CHANNEL_DATA (force or torque). The outputs of the network will indicate a classification of GOOD or BAD. Assign the outputs for the training data files a classification of GOOD.

(3) Scale inputs so they fall into the range 0 to 1.

(4) Reduce data set by eliminating points that are very close to other data points. This is a non-essential step done to reduce training time.

(5) Create "bad" data by making a grid covering the entire data space, then eliminating points either too close or too far from any GOOD data points. The "bad" data grid appears around the perimeter of the data in Figure 11. Assign these outputs a classification of BAD.

(6) Pad GOOD data by making a grid covering the entire data space, then eliminating points too far from any GOOD data points. The "good" data padding grid immediately surrounds the "good" training data in Figure 11. Assign these outputs a classification of GOOD.

(7) Run the network growing algorithm described in Section 2.3.

Note that the distances of steps 5 and 6 are selected empirically. The "good" data is padded so that its boundary lies halfway between the GOOD and the BAD data. Without padding of the GOOD data, the training algorithm may arbitrarily classify regions in the "unpopulated" area between the GOOD and the BAD regions as GOOD or BAD, despite the fact that a point may lie very close to data of the opposite type.

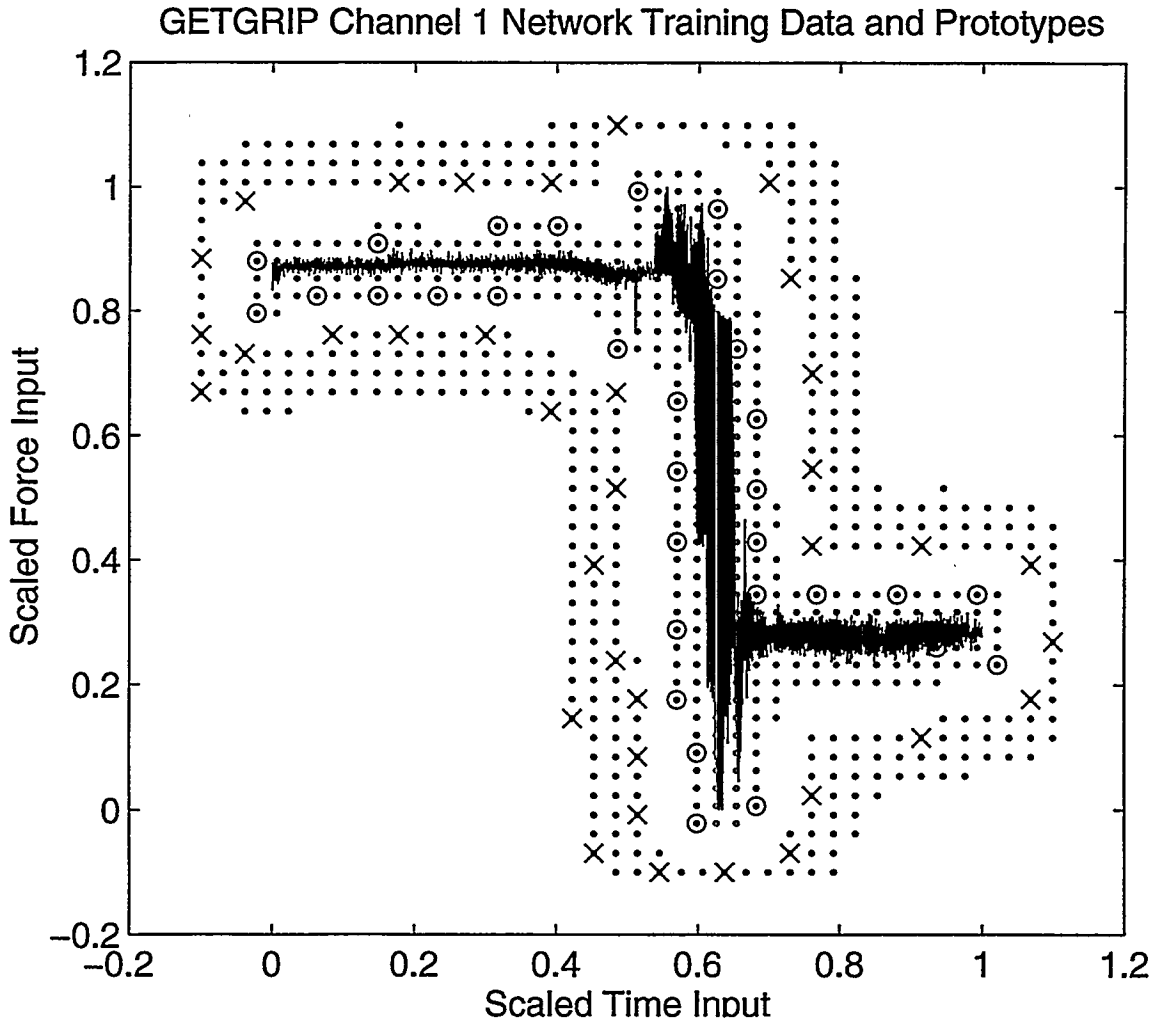


Figure 11 - The training data set for the GETGRIP channel 1 operation. "Good" data from the training files is shown as a connected line. Padding for the "good" data appears as a grid of points immediately surrounding the "good" data. Fabricated "bad" data forms the other grid surrounding and separated from the "good" data. Also shown are the prototypes selected by the training algorithm. 31 GOOD prototypes are shown as O's, and 36 BAD prototypes are shown as X's.

2.5 Results

2.5.1 Performance on training and test data

Naturally we desire that our failure detection system classify data from "good" operations as GOOD. Additionally, we would like the system to ignore infrequent data errors. Training data sets consisted of all of the "good" data, with "bad" training examples constructed as described in section 2.4. Test data included both "good" and "bad" runs.

The following tables summarize raw neural network results and shutdown results on both "good" and "bad" operations. Raw neural network classification on training data is not 100% due to the data set trimming described in section 2.4, step 4. The resulting very low classification error (0.03% for the getgrip operation) does not affect the shutdown behavior of the resulting system, however.

Classification rates for the "bad" test data are based on a human assessment, made by examining the force and torque signatures, of when a signal was "bad" or "good". For simplicity, only four conditions were identified: (1) always "good", (2) always "bad", (3) initially "good", became "bad" at time TB, and (4) initially "bad", became "good" at time TG. In reality, some of the data was more complex, for example starting out "bad", appearing "good" for a while, then becoming "bad" again. This complexity was ignored because it is irrelevant to the desired result of shutting down a process as soon as possible after it goes wrong. This fact affects the statistics of the raw neural network results, however. For example, on the test set of the GETCUT operation (Table 11), it appears that nearly 49% of the "bad" signatures were classified as GOOD. A more correct interpretation is that the signals, after being classified as BAD, returned to the "good" range and were classified as GOOD. Figure 12 examines an erroneous GETCUT operation in detail. Statistics such as these may also be produced when the errors are on the threshold of detection, as shown in Figure 13. The system performance is best assessed by the shutdown behavior rather than classification.

GETGRIP Operation. Raw Neural Network Results				
	Training Data Classification		Test Data Classification	
True State	GOOD	BAD	GOOD	BAD
GOOD	99.97%	0.03%	99.86%	0.14%
BAD	0.0%	0.0%	12.3%	87.7%

Table 1

GETGRIP Operation. Shutdown Results		
Shutdown criteria: more than 20% errors in 0.15 seconds		
File Description	Human Judgment/Desired Action	Shutdown Signal
11 Training files	Good Always/No Shutdown	No Shutdown
3 Good test files	Good Always/No Shutdown	No Shutdown
2 Bad test files	Bad from beginning/shutdown at start	Shutdown at 0.04 seconds

Table 2

GETWORK Operation. Raw Neural Network Results				
	Training Data Classification		Test Data Classification	
True State	GOOD	BAD	GOOD	BAD
GOOD	99.99%	0.01%	98.7%	1.3%
BAD	0.0%	0.0%	3.2%	96.8%

Table 3

GETWORK Operation. Shutdown Results		
Shutdown criteria: more than 20% errors in 0.15 seconds		
File Description	Human Judgment/Desired Action	Shutdown Signal
11 Training files	Good always/No shutdown	No Shutdown
1 Good test file	Good always/No shutdown	No Shutdown
1 Good test file	Good always/No shutdown	Shutdown at 0.0325 seconds †
4 Bad test files	Bad from beginning/Shutdown at start	Shutdown in 0.0325 to 0.04 second range
1 Bad test file	Bad at 29.25 seconds/Shutdown at 29.25 seconds	Shutdown at 30.35 seconds

Table 4

† This erroneous shutdown of a "good" operation was due to a transient in the first 0.5 seconds of the channel 3 signal. Data from a representative training operation and the flagged operation are shown in Figure 14. In a manufacturing or weapon disassembly environment, the operator would determine whether the operation was really erroneous or whether the shutdown signal was unwarranted. If the shutdown was unwarranted, this new data would be used to update the training set so that the erroneous shutdown signal will not occur again for similar timing sequences.

PUTXRAY Operation. Raw Neural Network Results				
	Training Data Classification		Test Data Classification	
True State	GOOD	BAD	GOOD	BAD
GOOD	100.0%	0.0%	100.0%	0.0%
BAD	0.0%	0.0%	45.2%	54.8%

Table 5

PUTXRAY Operation. Shutdown Results		
Shutdown criteria: more than 20% errors in 0.15 seconds		
File Description	Human Judgment/Desired Action	Shutdown Signal
11 Training files	Good always/No shutdown	No Shutdown
2 Good test files	Good always/No shutdown	No Shutdown
3 Bad test files	Bad from beginning/Shutdown at start	Shutdown at times 0.0325, 0.44, and 0.49 seconds

Table 6

GETXRAY Operation. Raw Neural Network Results				
	Training Data Classification		Test Data Classification	
True State	GOOD	BAD	GOOD	BAD
GOOD	100.0%	0.0%	99.93%	0.07%
BAD	0.0%	0.0%	35.0%	65.0%

Table 7

GETXRAY Operation. Shutdown Results		
Shutdown criteria: more than 20% errors in 0.15 seconds		
File Description	Human Judgment/Desired Action	Shutdown Signal
11 Training files	Good always/No shutdown	No Shutdown
2 Good test files	Good always/No shutdown	No Shutdown
2 Bad test files	Bad from beginning/Shutdown at start	Shutdown at 0.04 seconds
1 Bad test file	Bad at time 17 seconds/Shutdown at 17 seconds	Shutdown at 16.595 seconds

Table 8

PUTCUT Operation. Raw Neural Network Results				
	Training Data Classification		Test Data Classification	
True State	GOOD	BAD	GOOD	BAD
GOOD	99.9%	0.07%	99.99%	0.01%
BAD	0.0%	0.0%	43.0%	57.0%

Table 9

PUTCUT Operation. Shutdown Results		
Shutdown criteria: more than 20% errors in 0.15 seconds		
File Description	Human Judgment/Desired Action	Shutdown Signal
11 Training files	Good always/No shutdown	No Shutdown
2 Good test files	Good always/No shutdown	No Shutdown
3 Bad test files	Bad from beginning/Shutdown at start	Shutdown at 0.0325, 0.04, and 0.04 seconds
1 Bad test file	Bad at time 13 seconds/Shutdown at 13 seconds	Shutdown at 13.16 seconds

Table 10

GETCUT Operation. Raw Neural Network Results				
	Training Data Classification		Test Data Classification	
True State	GOOD	BAD	GOOD	BAD
GOOD	99.998%	0.002%	99.8%	0.2%
BAD	0.0%	0.0%	48.8%	51.2%

Table 11

GETCUT Operation. Shutdown Results		
Shutdown criteria: more than 20% errors in 0.15 seconds		
File Description	Human Judgment/Desired Action	Shutdown Signal
11 Training files	Good always/No shutdown	No Shutdown
2 Good test files	Good always/No shutdown	No Shutdown
2 Bad test files	Bad from beginning/Shutdown at start	Shutdown at 0.71 seconds
1 Bad test file	Bad at time 12.5 seconds/Shutdown at 12.5 seconds	Shutdown at 12.25 seconds
1 Bad test file	Bad at time 17 seconds/Shutdown at 17 seconds	Shutdown at 14.8675 seconds

Table 12

PUTWORK Operation. Raw Neural Network Results				
	Training Data Classification		Test Data Classification	
True State	GOOD	BAD	GOOD	BAD
GOOD	100.0%	0.0%	99.95%	0.05%
BAD	0.0%	0.0%	5.7%	94.3%

Table 13

PUTWORK Operation. Shutdown Results		
Shutdown criteria: more than 20% errors in 0.15 seconds		
File Description	Human Judgment/Desired Action	Shutdown Signal
11 Training files	Good always/No shutdown	No Shutdown
2 Good test files	Good always/No shutdown	No Shutdown
3 Bad test files	Bad from beginning/Shutdown at start	Shutdown at 0.0325, 0.04, and 0.04 seconds

Table 14

PUTGRIP Operation. Raw Neural Network Results				
	Training Data Classification		Test Data Classification	
True State	GOOD	BAD	GOOD	BAD
GOOD	100.0%	0.0%	99.996%	0.004%
BAD	0.0%	0.0%	31.4%	68.6%

Table 15

PUTGRIP Operation. Shutdown Results		
Shutdown criteria: more than 20% errors in 0.15 seconds		
File Description	Human Judgment/Desired Action	Shutdown Signal
11 Training files	Good always/No shutdown	No Shutdown
2 Good test files	Good always/No shutdown	No Shutdown
2 Bad test files	Bad from beginning/Shutdown at start	Shutdown at 6.49 and 6.26 seconds*

Table 16

** The delays in excess of 6.49 and 6.26 seconds are large compared with most of the results reported here. Examination of the data shows that the signal level differences between the "bad" and "good" data were small, near the threshold of detection.*

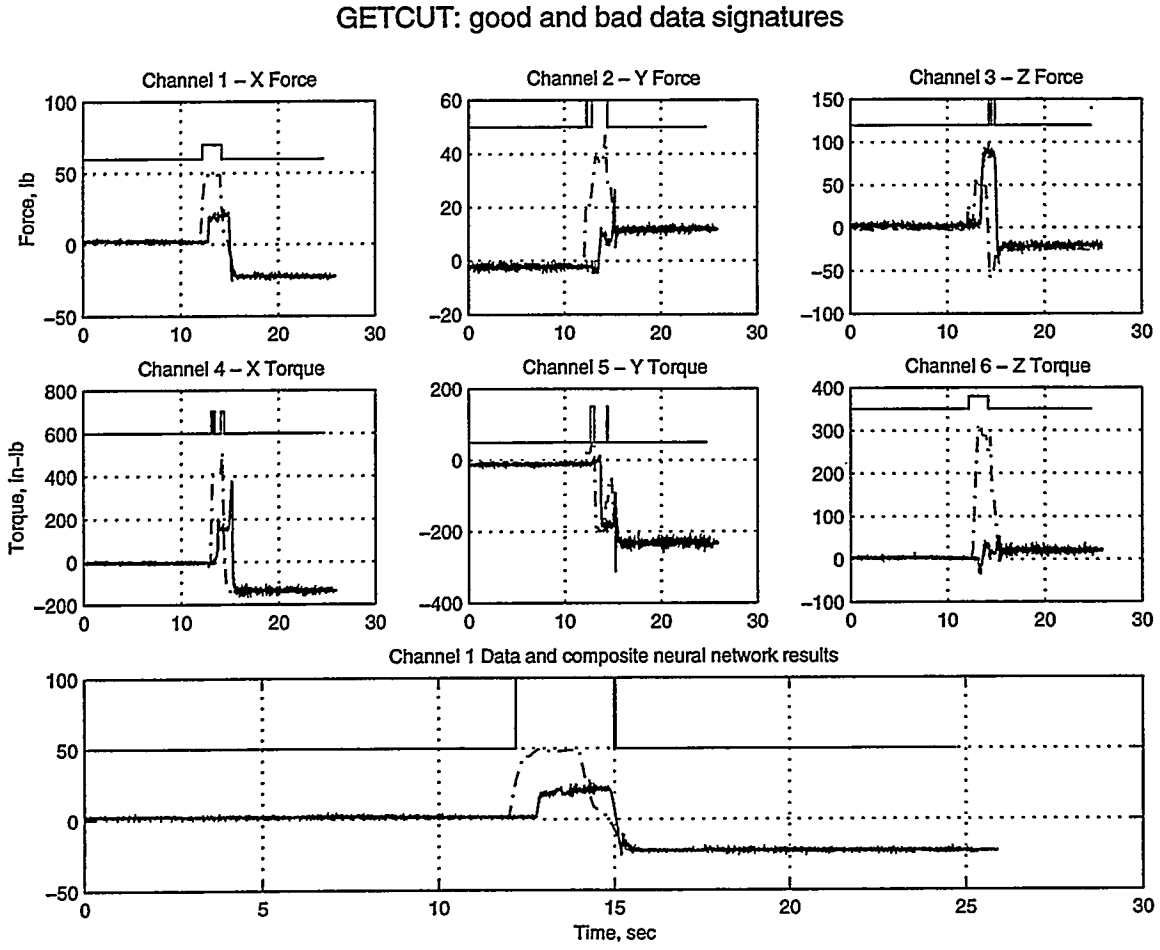


Figure 12 - A "bad" data operation is shown here as a dashed line in each graph, while the solid line is data from one of the training data sets. The binary signal at the top of each graph shows the neural network's evaluation of each signal. It is high when the data for the given channel is "bad". The bottom plot shows the composite error signal, formed by ANDing the 6 individual channel errors together.

The data were judged to be "bad" by a human at 12.5 seconds. The neural network/shutdown algorithm produced a shutdown signal at 12.25 seconds. The classification statistics shown in Table 17 were derived from this single operation. All data after approximately 15 seconds are considered to have a true state of BAD, even though the signals have returned to normal.

GETCUT Operation of Figure 12		
	Test Data Classification	
True State	GOOD	BAD
GOOD	97.8%	2.2%
BAD	79.4%	20.6%

Table 17

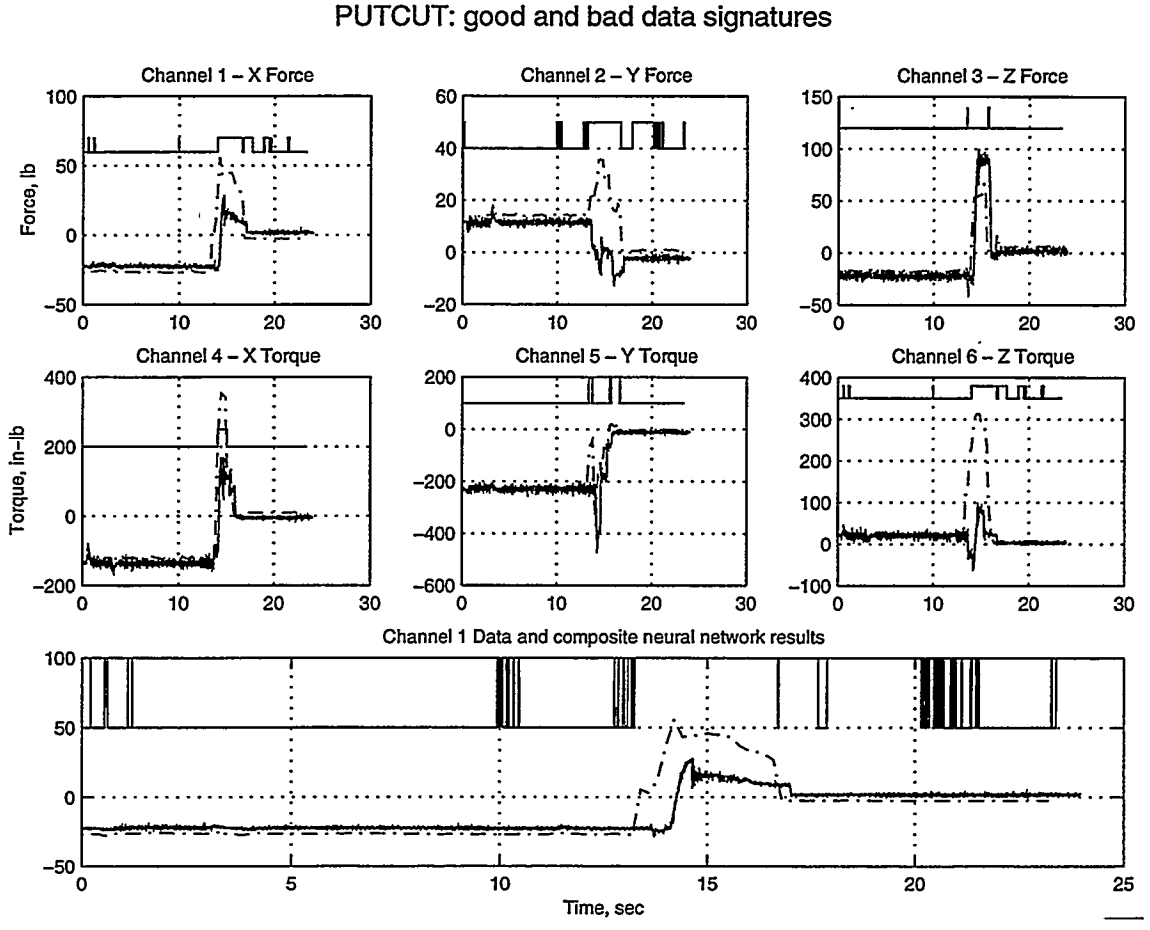


Figure 13 - This operation was judged to be "bad" from time 0 seconds to the end of the run. The neural network/shutdown algorithm produced a shutdown signal at time 0.04 seconds. Classification statistics for this run reflect the fact that the errors present here are often on the threshold of detection, as with channel 1 in the first seconds of the run. Statistics for this operation are shown below in Table 18.

Erroneous PUTCUT Operation of Figure 13		
	Test Data Classification	
True State	GOOD	BAD
GOOD	n/a	n/a
BAD	65.9%	34.1%

Table 18

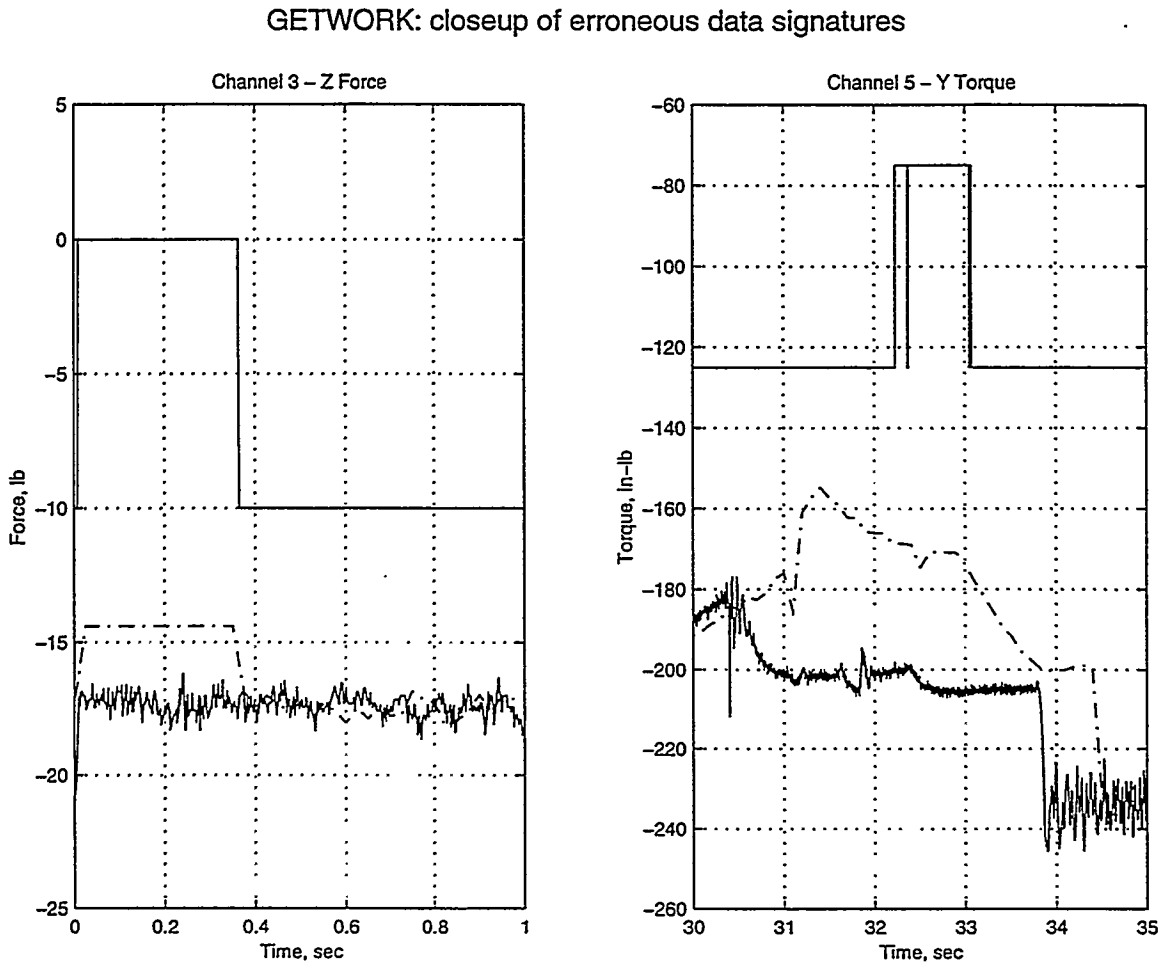


Figure 14 - This operation, presumed "good" by the robot operator, generated a shutdown signal at time 0.0325 seconds due to the transient on channel 3. Another anomaly on channel 5 would have shut this operation down in the 32-33 second range. See also Table 4 and its associated comments

To summarize the performance on the training and test data sets:

(1) The neural network failure detection system would have allowed all "good" operations to proceed without shutdown with one exception in the test set of the GETWORK operation. This case is examined in Table 4 and Figure 14.

(2) Most erroneous operations would have been shutdown within 0.1 second of the beginning of the error condition. Shutdown delay on "bad" operations of all eight types is summarized below.

Number of "bad" operations shutdown within given time of error condition	
Delay Time (seconds)	Number of operations
0.1	17
0.5	3
1.0	2
2.0	1
5.0	0
10.0	2
No shutdown	0

Table 19

2.5.2 Further error detection performance

The results described in Section 2.5.1 on the training and test data tell us that the error detection scheme does pass "good" operations and will appropriately generate a signal to shut down "bad" operations. But these results do not indicate how error detection performance and shutdown delay vary with error size. The following graphs (Figure 15 through Figure 30) reveal that information.

To generate each of these plots, the data for one "good" operation in the training set was selected. An error equaling a certain percentage of the signal range (for that operation only and for each individual channel) was added to the "good" data on all six channels. The data were processed as though the error condition began at each timestep from time T=0, to the end of the operation. For each start time, the error condition was presumed to persist to the end of the operation. Thus shutdown status and shutdown delay are determined versus time for each error size.

Taking the first graph, Figure 15, as an example, the dashed line indicates that shutdown would have occurred for a -5% error beginning at any time during the operation and persisting to the end of the operation. Shutdown delay is near zero

except around T=20 seconds, where a period of rapid transition precludes error detection (See Figure 3, also). The maximum delay to detection and shutdown for this error is 3 seconds and occurs at around 19 seconds into the operation. This indicates that there are 3 seconds during the rapid transition during which a -5% error would not be detected if the error did not persist beyond that 3 second time frame.

Figure 16 graphs shutdown rate versus error size, derived from the data of Figure 15. For example, the shutdown rate for a +1% error is about 37%. In Figure 15 we see that for a 1% error, detection will occur for errors beginning between times 0 and 12 seconds, and not for errors beginning between 12 and 32 seconds, which results in the 37% shutdown rate.

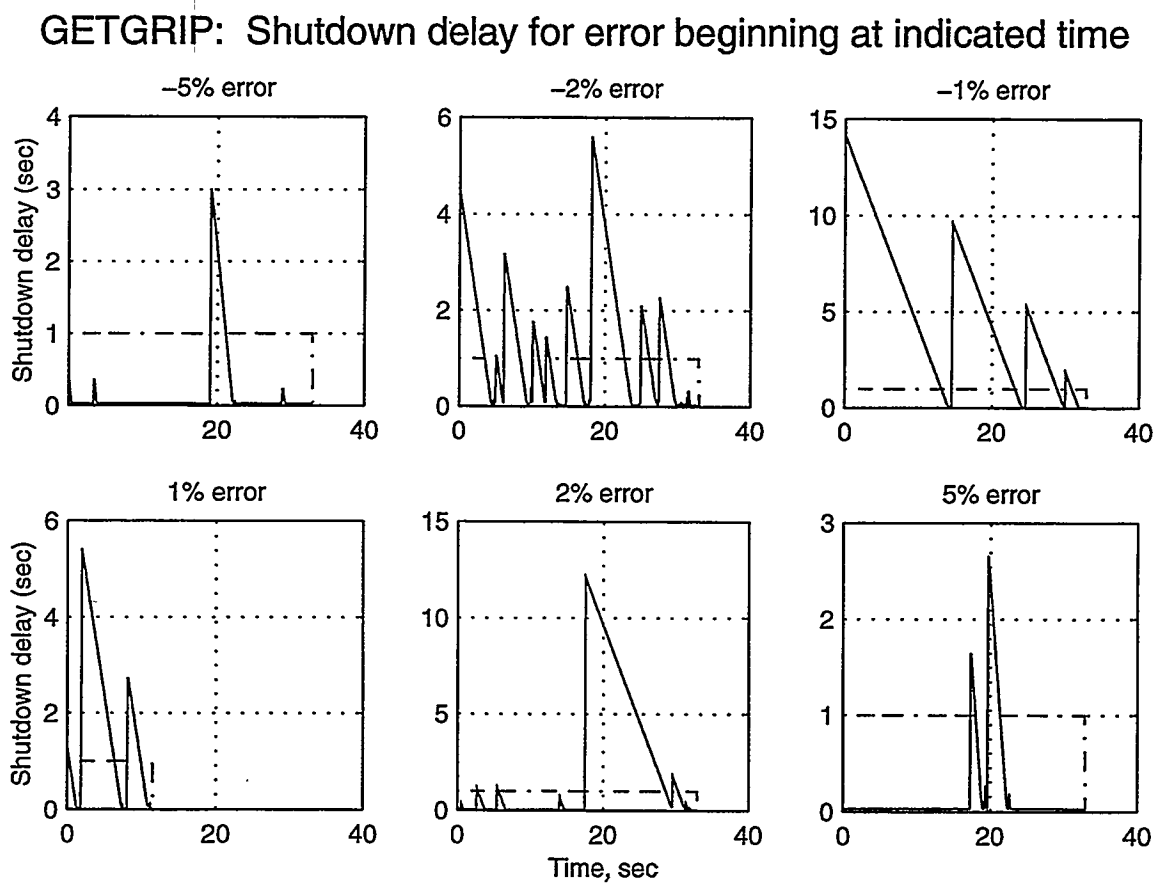


Figure 15 - Solid line indicates shutdown delay. Dashed line indicates shutdown status. Discussion in text.

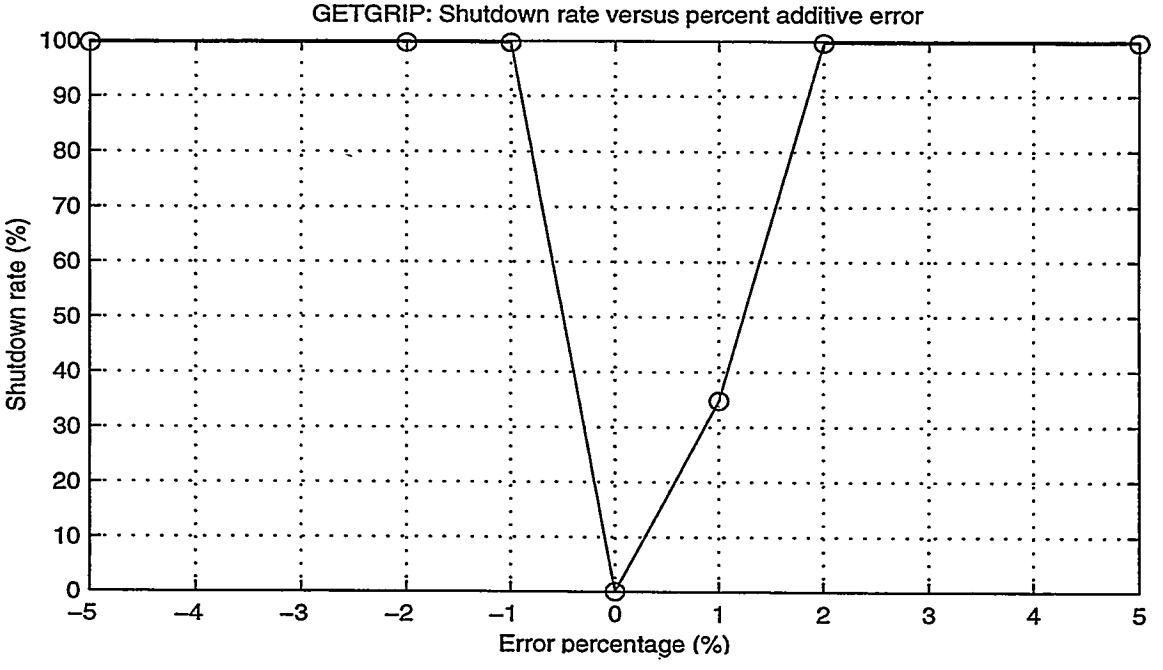


Figure 16 - Shutdown rate versus error percentage.

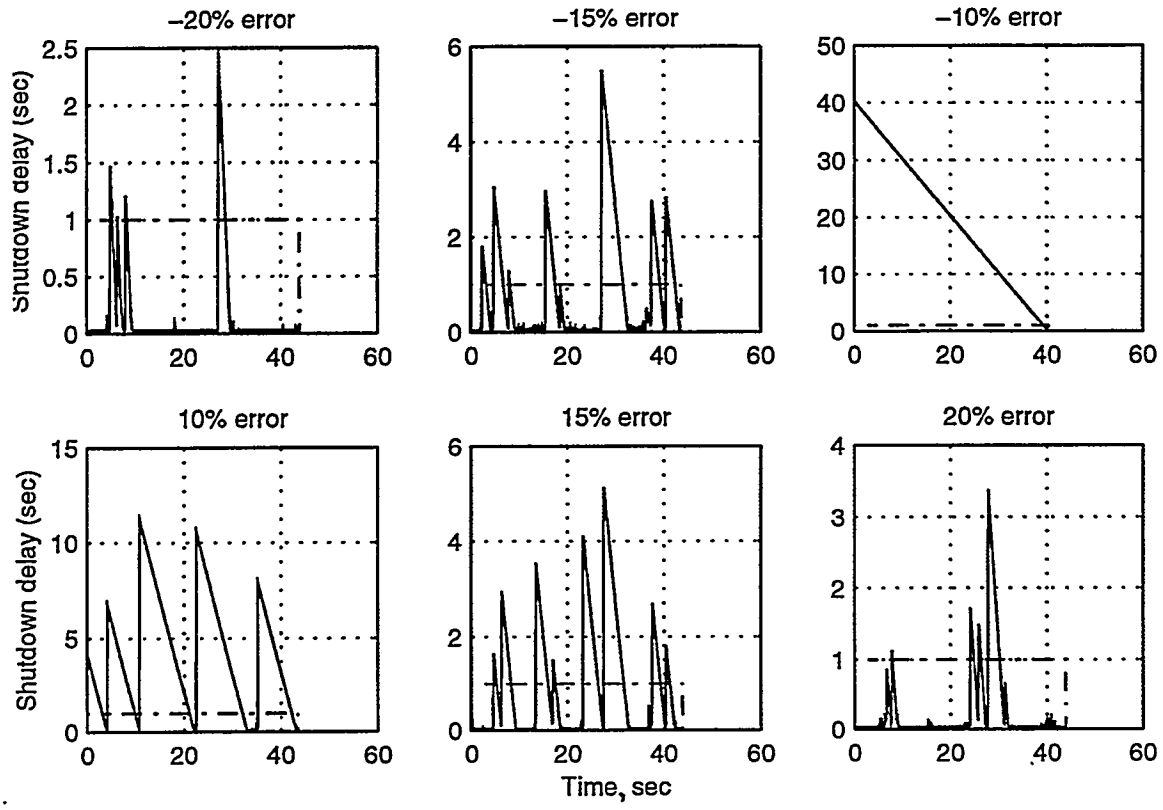


Figure 17 - Solid line indicates shutdown delay. Dashed line indicates shutdown status.

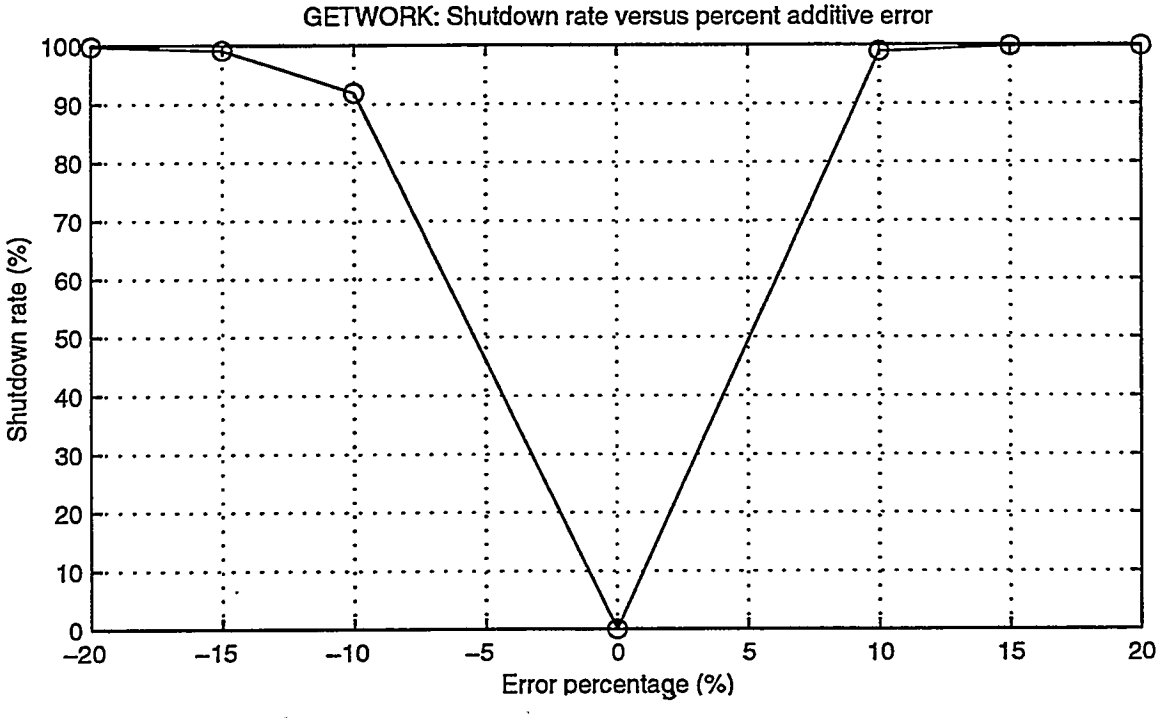


Figure 18 - Shutdown rate versus error percentage.

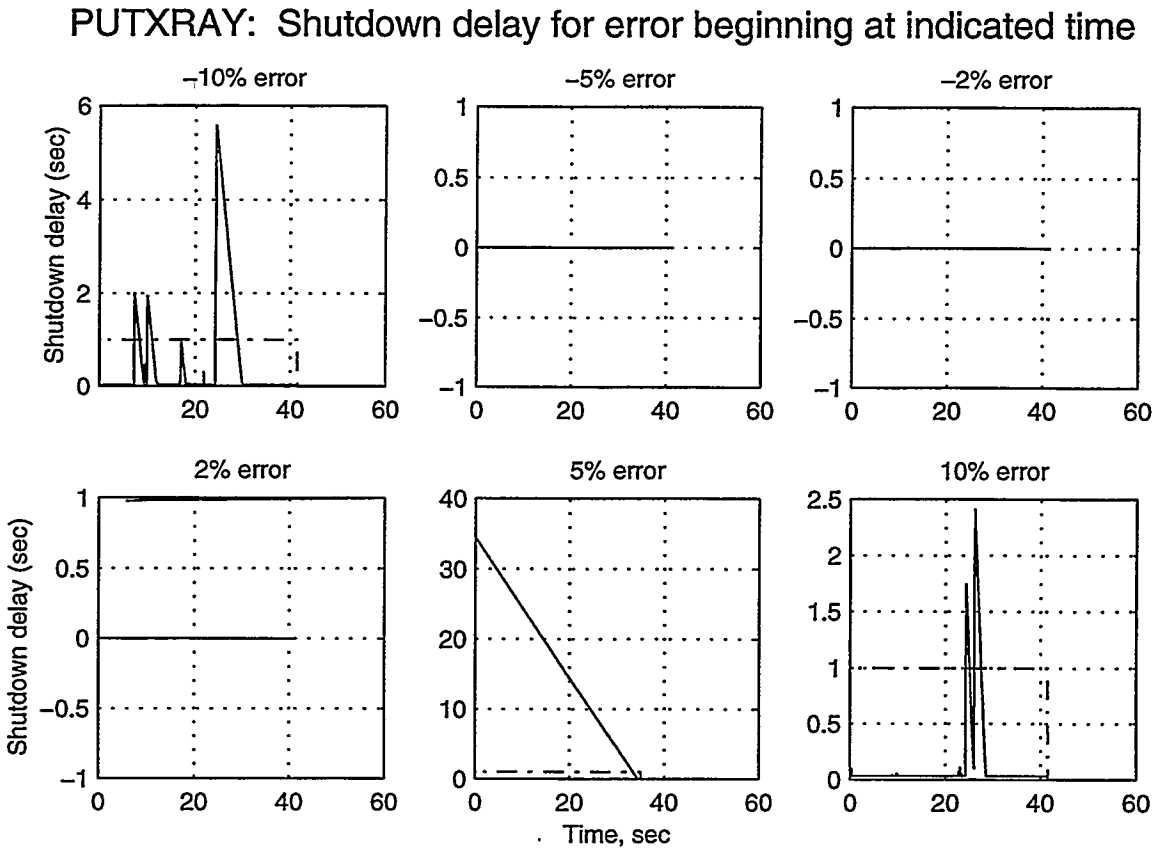


Figure 19 - Solid line indicates shutdown delay. Dashed line indicates shutdown status.

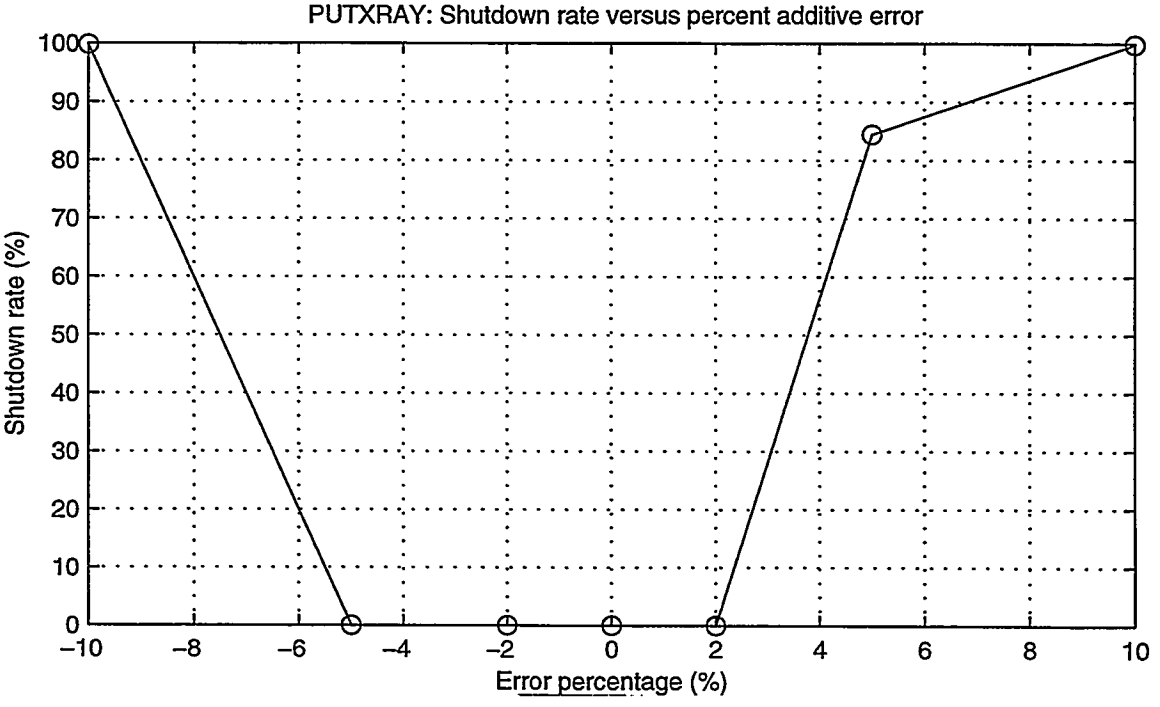


Figure 20 - Shutdown rate versus error percentage.

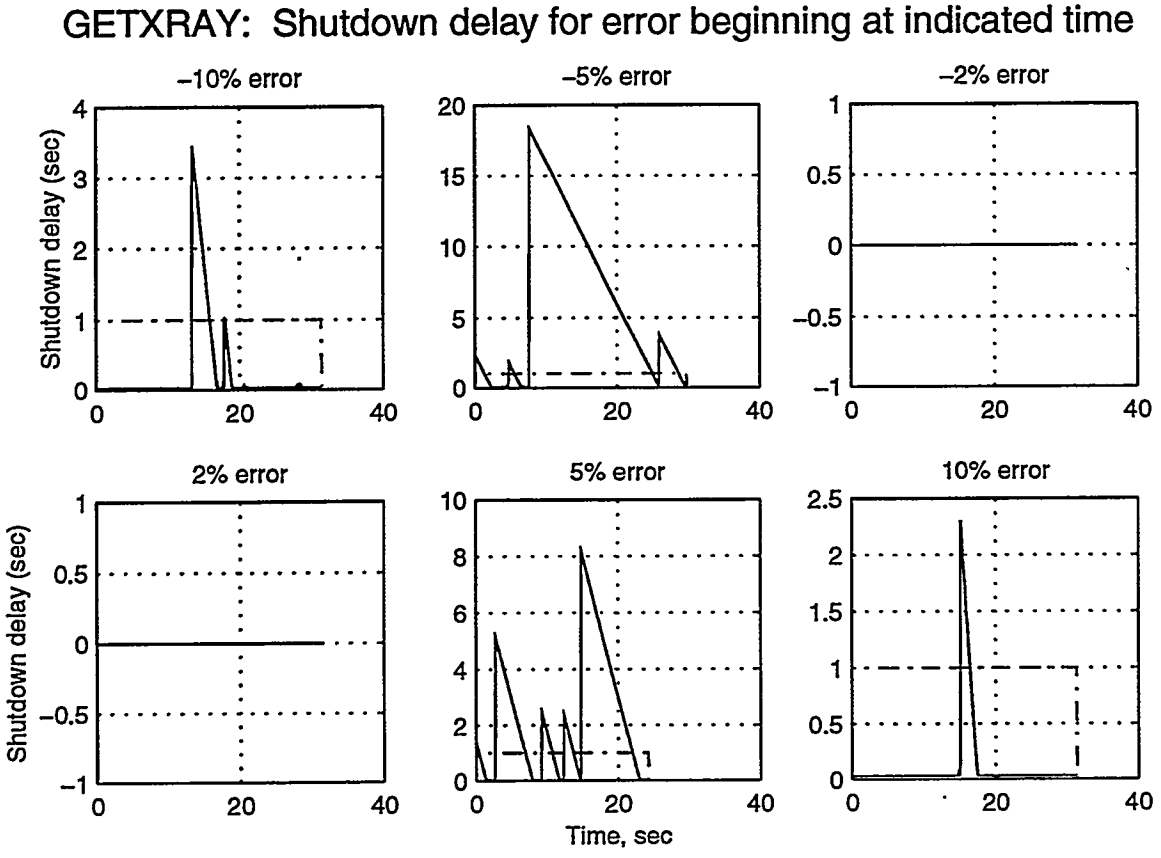


Figure 21 - Solid line indicates shutdown delay. Dashed line indicates shutdown status.

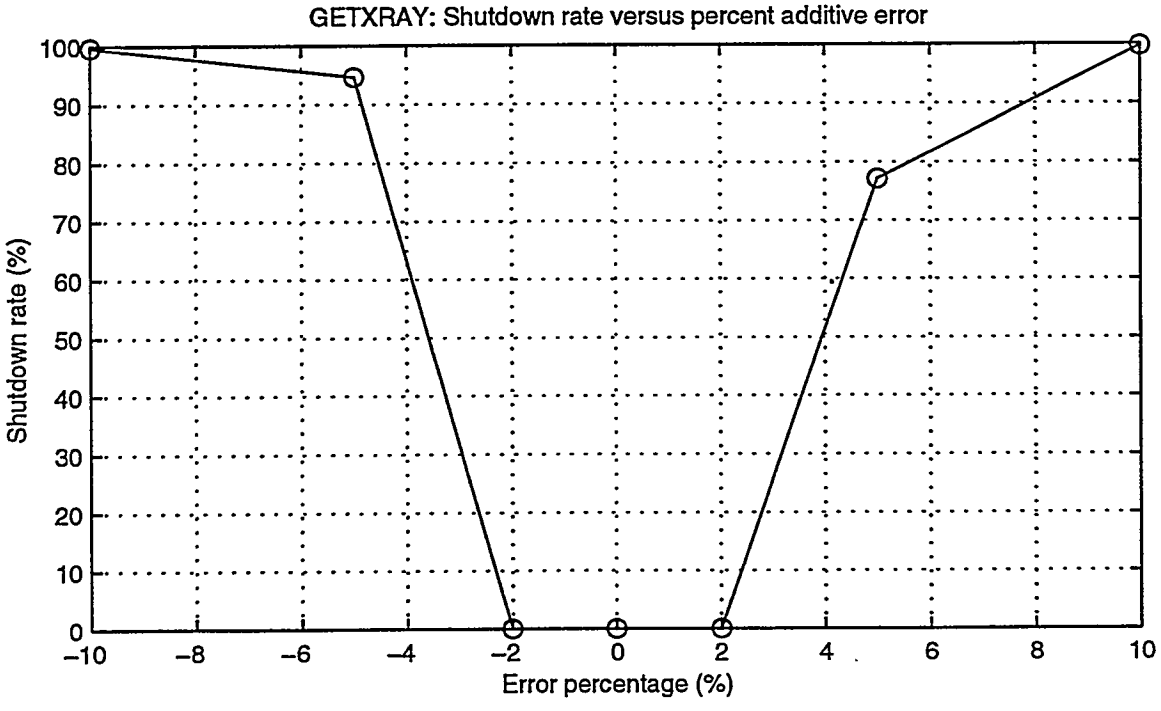


Figure 22 - Shutdown rate versus error percentage.

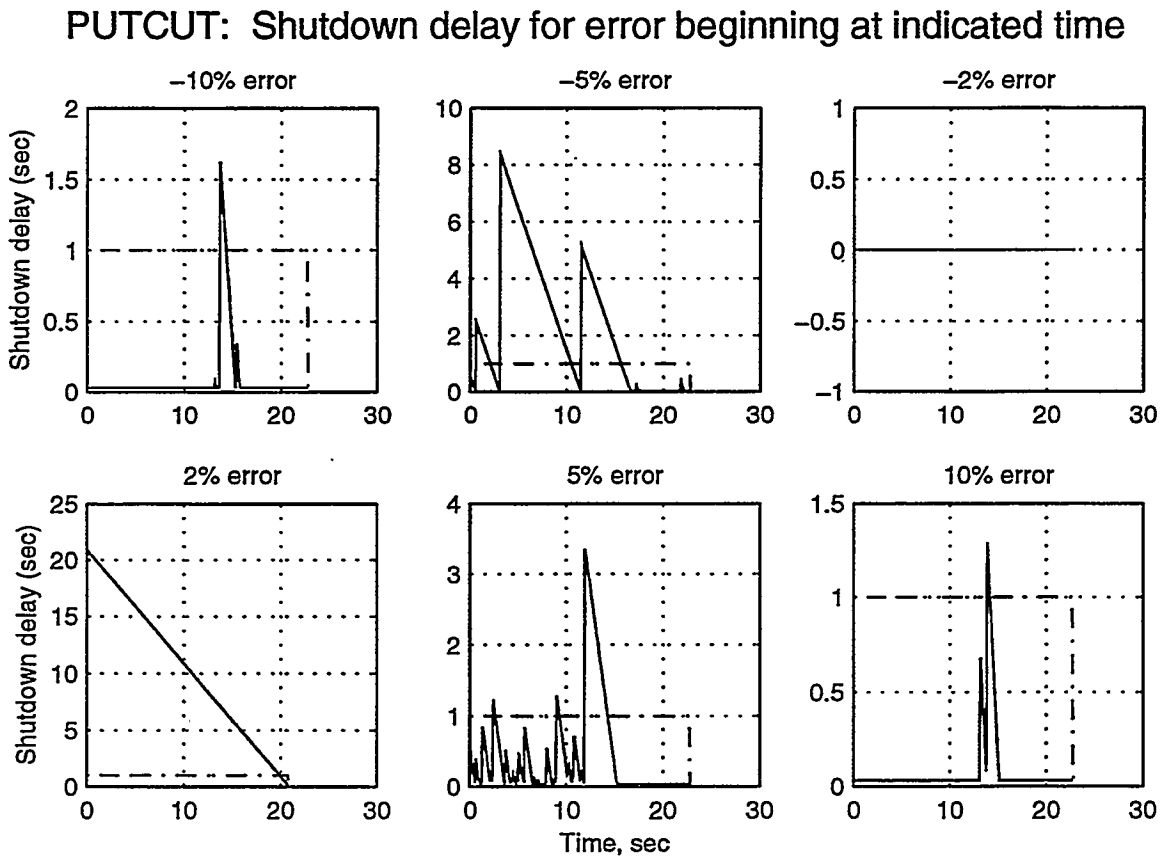


Figure 23 - Solid line indicates shutdown delay. Dashed line indicates shutdown status.

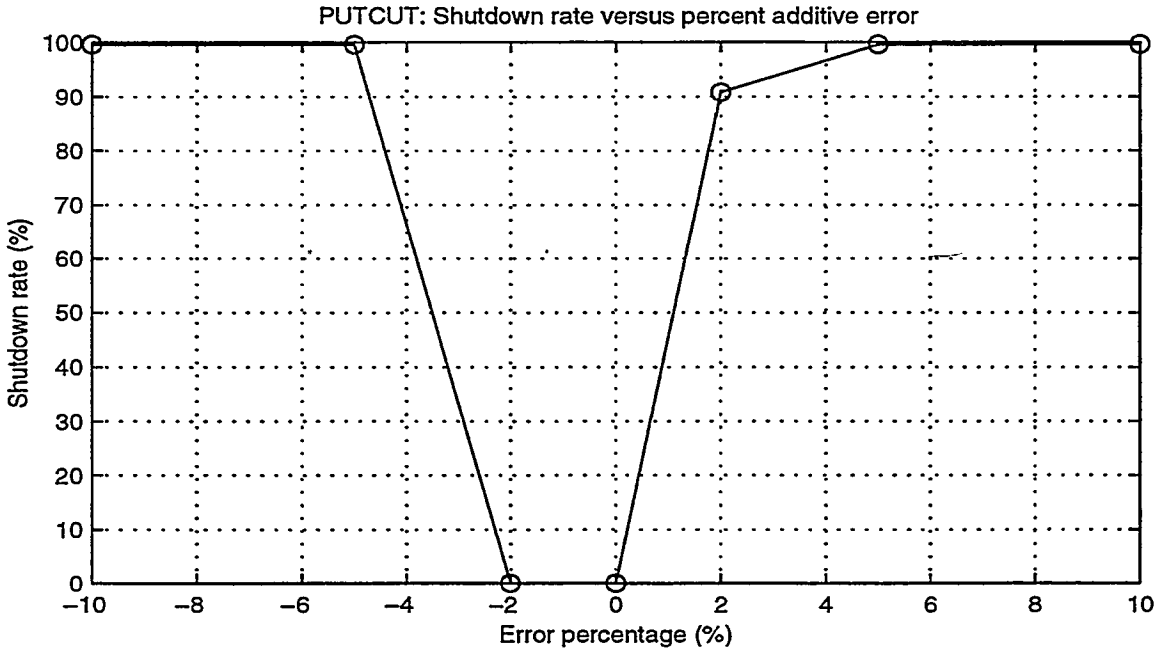


Figure 24 - Shutdown rate versus error percentage.

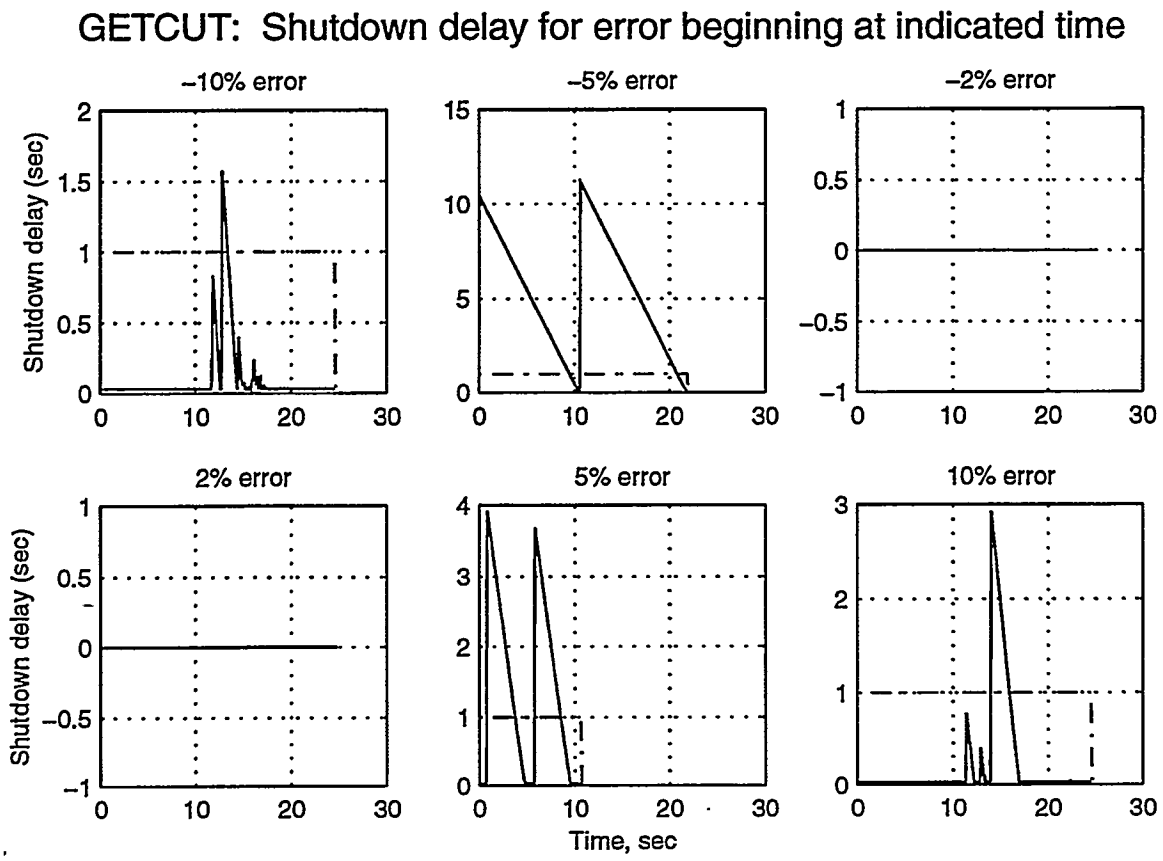


Figure 25 - Solid line indicates shutdown delay. Dashed line indicates shutdown status.

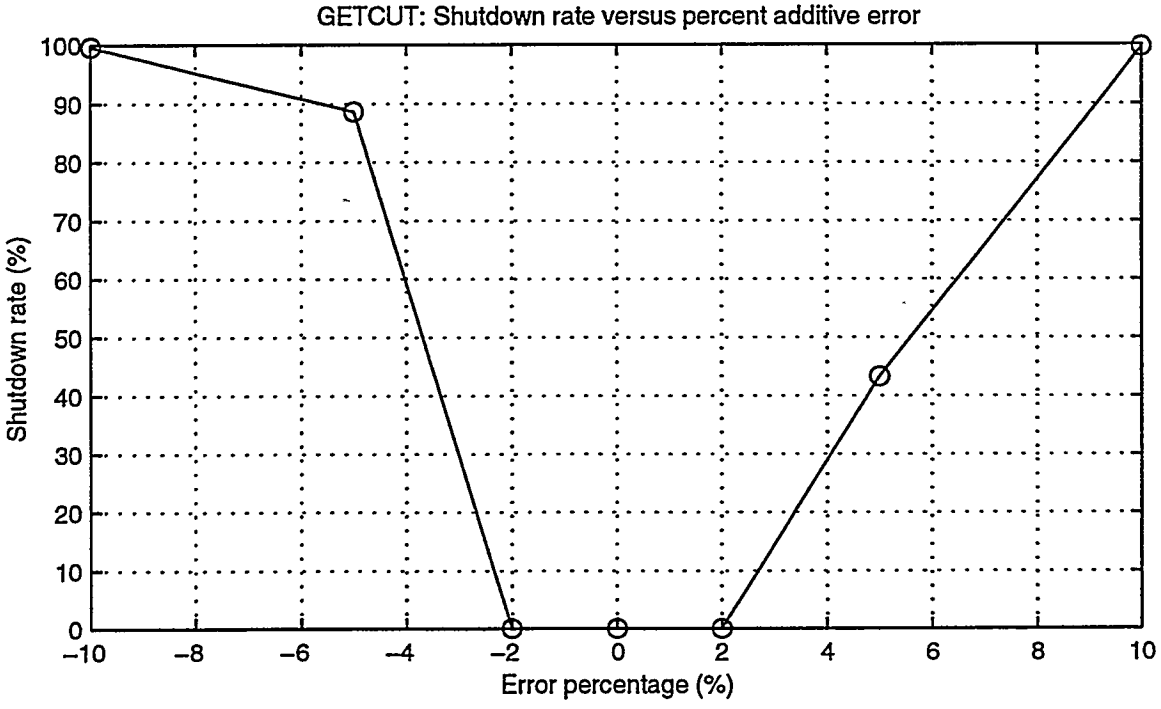


Figure 26 - Shutdown rate versus error percentage.

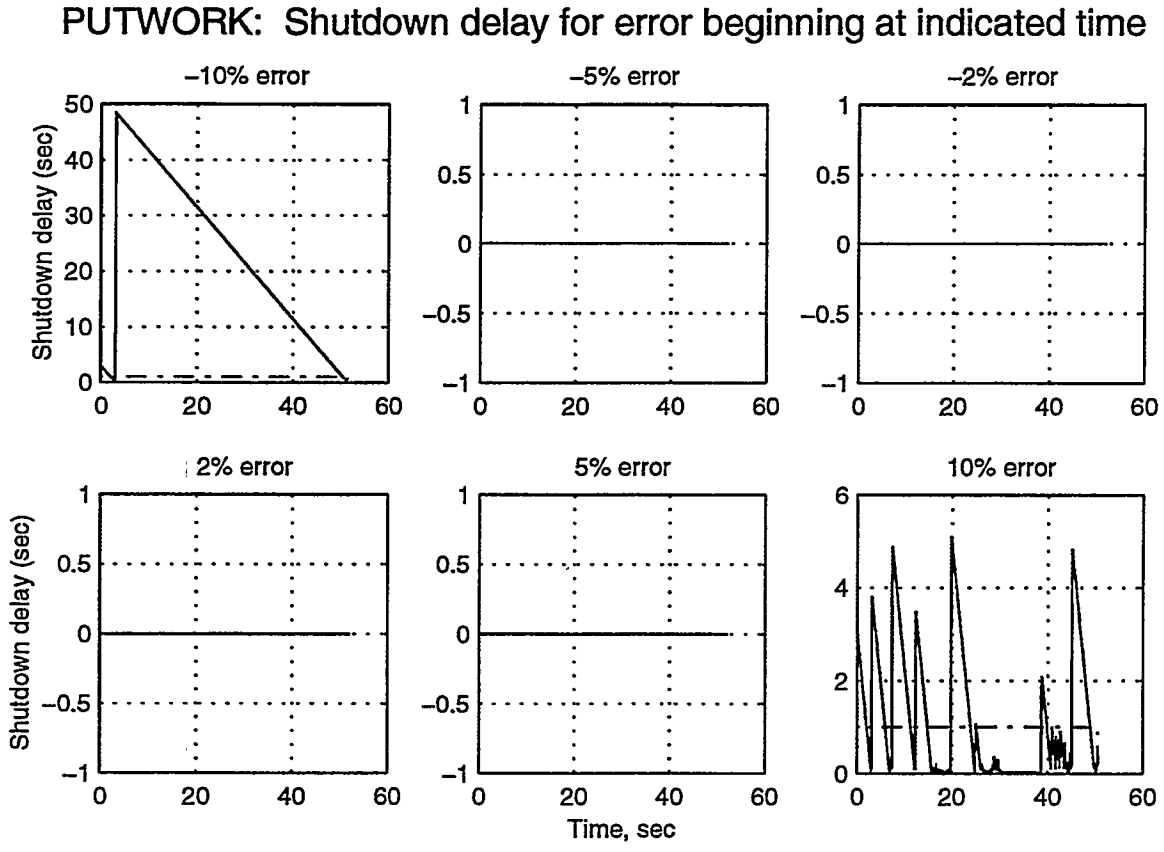


Figure 27 - Solid line indicates shutdown delay. Dashed line indicates shutdown status.

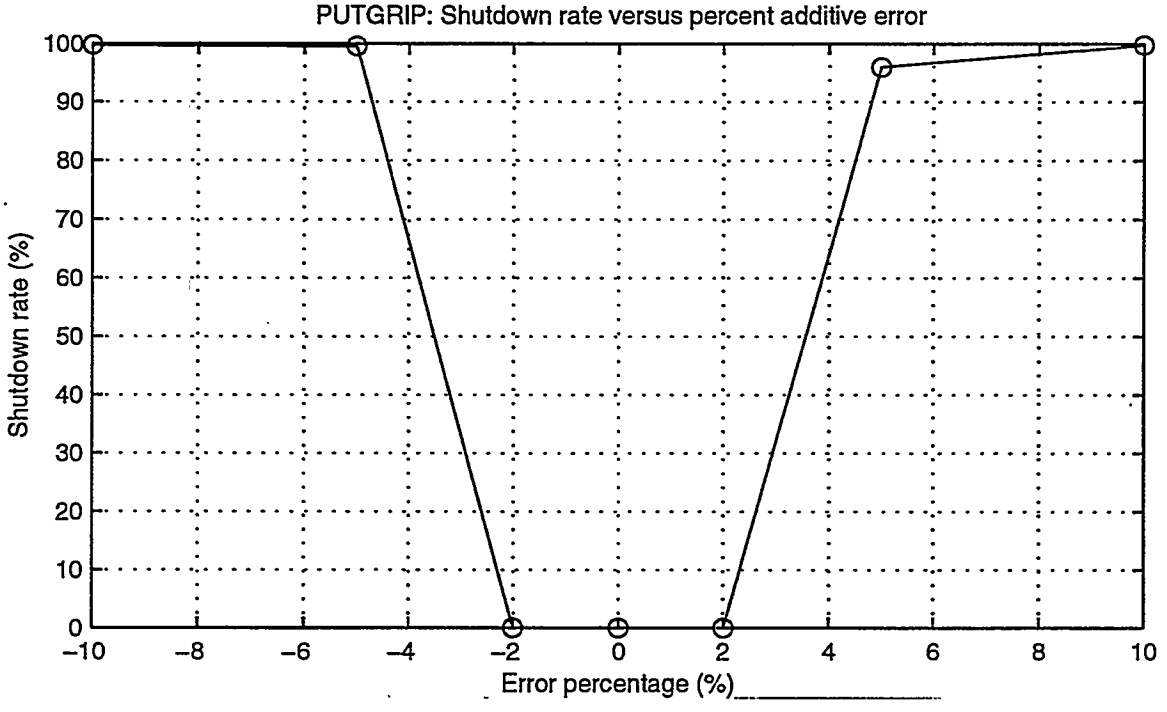


Figure 28 - Shutdown rate versus error percentage.

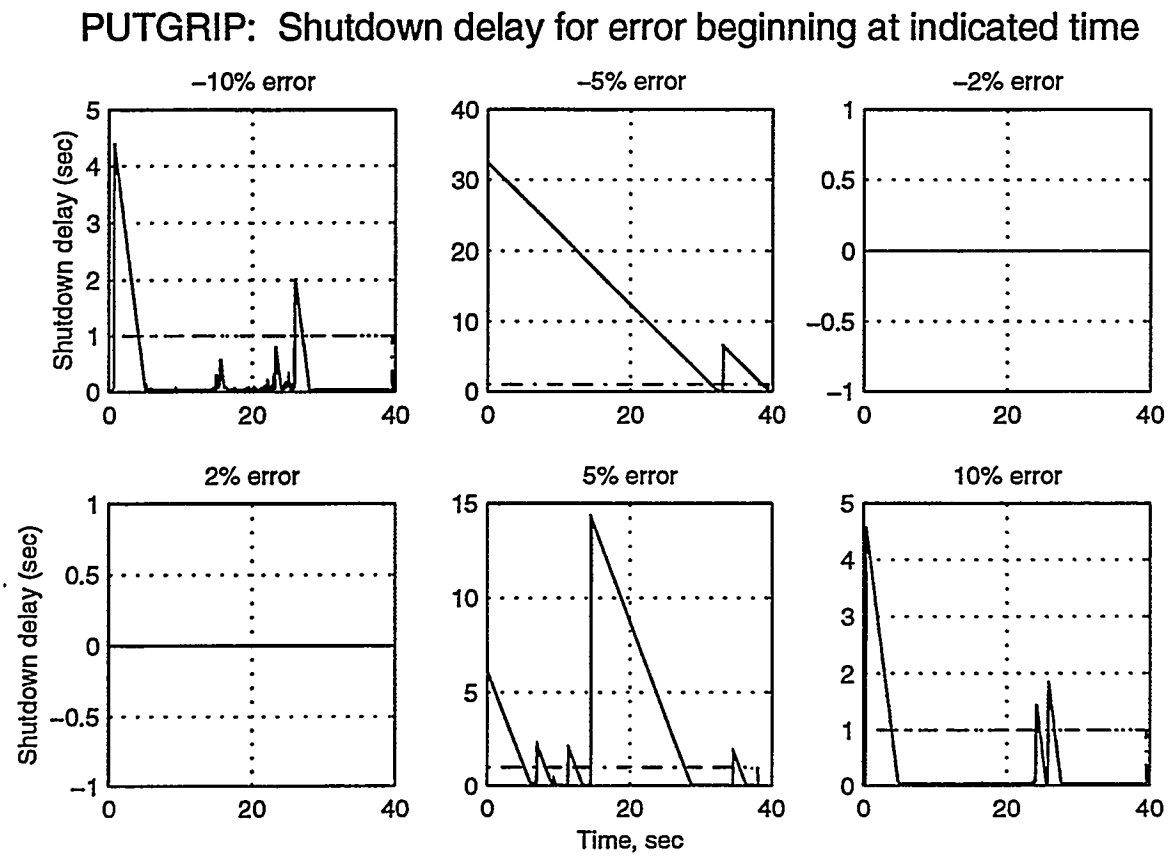


Figure 29 - Solid line indicates shutdown delay. Dashed line indicates shutdown status.

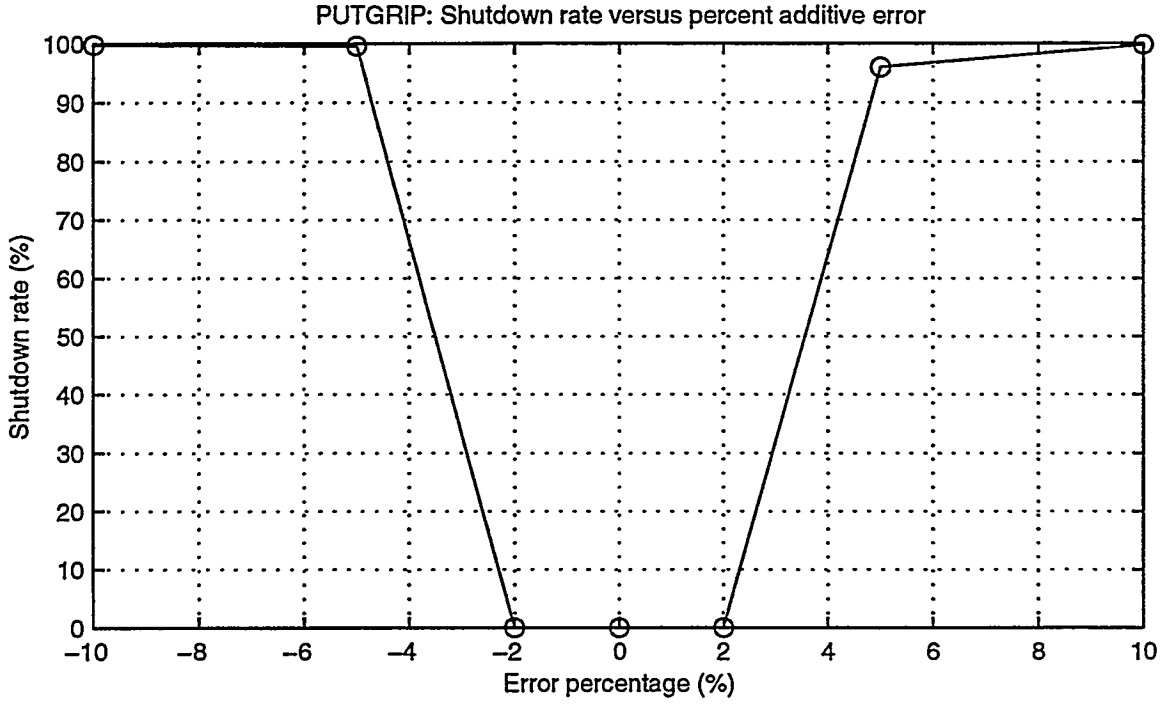


Figure 30 - Shutdown rate versus error percentage.

2.6 Computational Requirements

A non-optimized implementation of the networks running under Matlab was used to implement these networks. The following tables summarize the number of prototype nodes generated for each operation and channel, as well as the number of floating point operations used to execute each network for one input vector.

Nodes per Network						
Operation	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
GETGRIP	67	71	115	71	78	68
GETWORK	71	86	73	79	59	74
PUTXRAY	67	71	73	77	74	62
GETXRAY	75	69	82	76	73	70
PUTCUT	72	68	71	83	88	85
GETCUT	64	72	65	66	71	73
PUTWORK	71	72	80	67	82	71
PUTGRIP	74	73	70	85	71	94

Table 20 - Number of Nodes

Computation Requirements for Neural Networks (floating point operations per cycle)							
Operation	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	All
GETGRIP	807	855	1383	835	939	819	5658
GETWORK	855	1035	879	951	711	891	5322
PUTXRAY	807	855	879	927	891	747	5106
GETXRAY	903	831	987	915	879	843	5358
PUTCUT	867	819	855	999	1059	1023	5622
GETCUT	771	867	783	795	855	879	4950
PUTWORK	855	867	963	807	987	855	5334
PUTGRIP	891	879	843	1023	855	1131	5622

Table 21 - Computational Requirements

The total number of floating point operations for all six channels is the greatest for the GETGRIP operation, with 5658 flops per evaluation of the network. This would require a computer with 0.566 megaflops (million floating point operations per second) capability to keep up with a 100 Hz data rate, or 2.263 megaflops to keep up with a 400 Hz data rate.

3. Results from Deburring Workcell

This project was initiated using data from a robot deburring operation. Most of the signal variation occurred in only one channel of that operation (Z Force), although other channels were thought to carry useful information, especially in an error situation. A typical operation is plotted in Figure 31. Data collected for this investigation is summarized in Table 22.

Several approaches were investigated and abandoned; the most informative are summarized in this section. "Bad" data for each of these cases was generated in a different manner than in the work of Section 2. For these other approaches, "bad" data was generated by corrupting "good" data in a number of fashions, as tabulated in each section below.

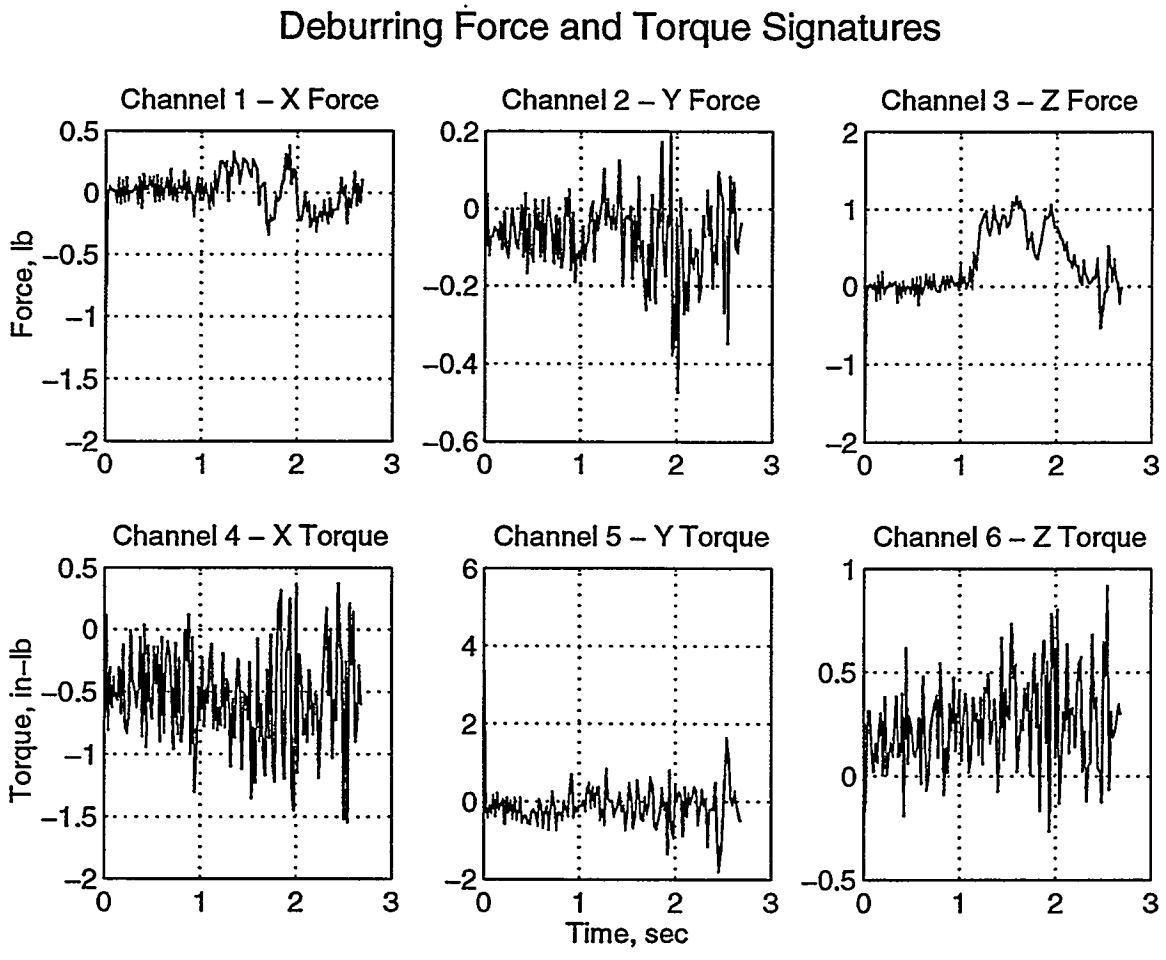


Figure 31 - Deburring Operation Force and Torque Signatures

Data Collected for Deburring Operation	
Data	Comment
70 GOOD training operations	60 used for training, 10 for test
4 operations in which tool was not enabled	Pneumatic tool not energized
4 operations in which airflow was restricted to tool	Pneumatic tool not fully energized
3 brush liftoff operations	Contact prematurely lost during brush operation

Table 22

This approach of generating "bad" data by corrupting "good" data has its advantages and disadvantages. The primary advantage is that it yields the ability to train networks with diagnostic capability. There were several disadvantages. First, this process resulted in a tremendous amount of data (hundreds of megabytes) being generated in order to provide the variety of error conditions necessary to a rich training set. This resulted in long preprocessing times and a limited ability for experimentation because of the long turnaround times. Often human judgment was required to specify when data was "good" and "bad" for the training and test data. This, too, was very time consuming. Consequently, when the focus of the work moved from the deburring operation to the material handling operations, this data breaking technique was abandoned.

3.1 Backpropagation network with general input set

The first approach taken was to design a network with a wide variety of preprocessed inputs and an ambitious output set. The appeal of this approach is that it is very general and could be applied to a wide variety of data signals without customization. The input and output sets are shown in Table 23 and Table 24.

Some "bad" data were generated for this problem by collecting data under certain known error conditions. These have been summarized in Table 22. Additionally, "bad" data were generated by "data breaking", or deliberately corrupting "good" data files. In Table 25, the column "Data Breaking Technique" tells what was done to the "good" data to create "bad" data, and the "Error" column describes the diagnosis that the network might be expected to generate.

Inputs and Definition for Backpropagation Network	
Inputs	**Definition**
(1) PHASE_TIME	Time in current phase
(2) FIND_EDGE	Current operation is a "find edge" operation
(3) BRUSH_EDGE	Current operation is a "brush edge" operation
(4) TRANSIENT_PHASE	Current operation is in transient phase
(5) RETRY	Current operation is a retry attempt
(6-8) IFORCEX, IFORCEY, IFORCEZ	Integrated X, Y, and Z force signals
(9-11) ITORQX, ITORQY, ITORQZ	Integrated X, Y, and Z torque signals
(12-23) FORCE_MAG1, FORCE_MAG2,... FORCE_MAG12	Magnitude of Force vector for last 12 time steps
(24-35) TORQ_MAG1, TORQ_MAG2,... TORQ_MAG12	Magnitude of Torque vector for last 12 time steps
(36-47) ZFORCE1, ZFORCE2,...ZFORCE12	Z force for last 12 time steps
(48-55) FX_FFT_1, FX_FFT_2,...FX_FFT_8	8 FFT values for X force signal
(56-63) FY_FFT_1, FY_FFT_2,...FY_FFT_8	8 FFT values for Y force signal
(64-71) FZ_FFT_1, FZ_FFT_2,...FZ_FFT_8	8 FFT values for Z force signal
(72-79) TX_FFT_1, TX_FFT_2,...TX_FFT_8	8 FFT values for X torque signal
(80-87) TY_FFT_1, TY_FFT_2,...TY_FFT_8	8 FFT values for Y torque signal
(88-95) TZ_FFT_1, TZ_FFT_2,...TZ_FFT_8	8 FFT values for Z torque signal

Table 23

Outputs and Definition for Backpropagation Network	
Outputs	Definition
(1) GOOD	True if status for operation at current time is GOOD
(2) NOISY	Some signal/signals is/are noisy
(3) DEAD	Some signal is stuck at constant value
(4-9) NOISY_FX, NOISY_FY, NOISY_FZ, NOISY_TX, NOISY_TY, NOISY_TZ	Indicated signal is noisy
(10-15) DEAD_FX, DEAD_FY, DEAD_FZ, DEAD_TX, DEAD_TY, DEAD_TZ	Indicated signal is stuck at a constant level
(16) BRUSH_LIFTOFF	Brush has lifted when it shouldn't during BRUSH operation
(17) SUCCESSFUL_BRUSH	Brush operation proceeding successfully
(18) FIND_OFF_CENTER	Sensor cluster is off-center during FIND operation
(19) SUCCESSFUL_FIND	Find operation proceeding successfully
(20) EXCESSIVE_FORCE	Force is higher than expected
(21) WRONG_PROFILE	Signature of the data isn't right. E.G. - BRUSH operation signature present when FIND input is true
(22) OFF_THE_WALL	Data appears to be random
(23) TOOL_NOT_ENERGIZED	Pneumatic tool is not under power
(24) RESTRICTED_AIR	Restricted airflow to pneumatic tool

Table 24

"Data Breaking" and Output Error for Backpropagation Network	
Data Breaking Technique	Error
Additive Noise	NOISY, NOISY_XX, where XX indicates all affected channels
Channel stuck at constant value	DEAD, DEAD_YY, where YY indicates all affected channels

Table 25

A data set with 16833 data points (2441 "good" and 14392 "bad") derived from the collected data and "broken" data was divided into test and training sets with roughly 80% of the data forming the training set. A variety of fully connected feed-forward networks trained using backpropagation were trained and tested, all proving to have poor generalization. The reason for this, we believe, is that despite the large training set, it was not large enough to train a network with sufficient complexity to handle this problem. For example, a fully connected feed-forward network with 95 inputs, 2 hidden layers of 15 units apiece, and 24 outputs will have 39,960 weights, or free parameters. The number of free parameters exceeds the number of training points by a factor of 3. A common rule of thumb is that the number of unique training points must equal at least the number of free parameters to achieve "good" generalization. This condition is not met in this case, even with a data set that is already ponderously large and time-consuming to pre-process. Additionally, large networks are notoriously difficult to train to the accuracy of which they are capable. It became clear that a simpler set of neural network inputs must be derived from the data.

3.2 General Regression Neural Network with Carefully Selected Inputs

In this approach, the data for the deburring operation was carefully examined for features that characterized it. Algorithms were written to extract these features for use as the neural network inputs, summarized in Table 26. The find and brush phases of this operation were not separated for this study as they were in Section 3.2, so these phases are not represented in the input or output data sets. Both backpropagation and General Regression Neural Networks (GRNN) were developed for a subset of this data, with the GRNN architecture performing slightly better than the backprop networks. Consequently the rest of the work was carried out using GRNNs.

GRNN Neural Network Inputs		
Input		Description
(1)	SADFZ_1040	Sum of absolute values of Z force (FZ) during early portion of data (a noise measure)
(2)	SAD_1040	Sum of absolute values of all channels during early portion of data (a noise measure)
(3)	PLUS_EDGE	Count of positive edges in FZ data
(4)	NEG_EDGE	Count of negative edges in FZ data
(5)	SAMPLES	Number of samples in data profile
(6)	MC_HIGH	Maximum consecutive number of high entries in FZ data after conversion to square wave
(7)	MC_LOW	Maximum consecutive number of low entries in FZ data after conversion to square wave
(8)	HIGH_AREA	Area under MC_HIGH portion of FZ

Table 26

Data breaking techniques were used to create "bad" data for this network. A unique output for the network was assigned to each data breaking technique as well as for each type of collected "bad" data (Table 22). The network outputs are summarized below in Table 27. It was expected that there would be some confusion between the diagnostic outputs during some cases of erroneous inputs, but that the diagnosis would be useful nonetheless. This did prove to be the case, as shown in Table 28 and Table 29.

GRNN Neural Network Data Outputs/Data Breaking Techniques	
Output/Data Breaking Technique	Description
(1) GOOD	No error in data
(2) NOISY *	Data is excessively noisy
(3) DEAD *	Data signal is flat
(4) RANDOM *	Data signal is random
(5) TOOL_NOT_ENABLED	Pneumatic tool is not under power
(6) RESTRICTED_AIR	Restricted airflow to pneumatic tool
(7) SHORT_FIND *	Contact occurred too soon during FIND operation. Possible collision with object in environment
(8) TRIPLE_AMP *	Signal amplitude is too large
(9) THIRD_AMP *	Signal amplitude is too small
(10) EXPTIME *	Time-scale for operation is expanded. Operation taking too long
(11) CMPTIME *	Time-scale for operation is compressed. Operation not taking enough time
(12) CHOPPED *	Signal has been chopped into 0.25 second chunks and scrambled
(13) OTHER *	Preprocessed data near extents of input signal range

*Table 27 - * Indicates data breaking technique as well as a network output.*

A separate network was designed for each time step from 0.25 seconds onward, in 0.25 second steps, and there was a final network designed to be executed when the operation was complete. During operation, the data window from 0 seconds to the current time would be preprocessed every quarter second and the data fed to the appropriate neural network.

Desired Classification	Performance of GRNN Neural Network on Training Data Time == Operation Complete Number of Actual Classifications by Neural Network												
	GOOD	NSY	DEAD	RND	TNE	RES	SHRT	TRPL	THRD	EXPT	CMPT	CHOP	OTHER
GOOD	60												
NOISY		18											
DEAD			15										
RANDOM				60									
TNE					4								
RES	5					3							
SHORT							60						
TRIPLE								30					
THIRD									30				
EXPTIME										30			
CMPTIM							2				28		
CHOP							3				1	26	
OTHER													256

Table 28

Desired Classification	Performance of GRNN Neural Network on Test Data Time == Operation Complete Number of Actual Classifications by Neural Network												
	GOOD	NSY	DEAD	RND	TNE	RES	SHRT	TRPL	THRD	EXPT	CMPT	CHOP	OTHER
GOOD	10												
NOISY		1										1	1
DEAD			3										
RANDOM				9								1	
TNE					1								
RES	1												
SHORT							9						
TRIPLE								9					
THIRD									9				1
EXPTIME						1				9			
CMPTIM											10		
CHOP							1					8	1
OTHER													

Table 29

The classification matrix for the final network operating on the training and test data sets are shown in Table 28 and Table 29. The rows and columns of Table 28 are labeled with abbreviated names of the neural network outputs. Each row indicates a desired data classification, while each column indicates the actual classification that the neural network produced. Zero valued entries in the table are left blank for clarity. The first row of Table 28, for example, tells us that of 60 "good" entries in the training data set, 60 were classified as GOOD. Likewise, 18 noisy entries of 18 were correctly classified as NOISY. Ideally then, all entries in the table would show up on the primary diagonal of the table. On row 6, the restricted air entry indicates that 5 of 8 restricted air entries were classified as GOOD, while the remaining 3 were correctly classified. Comparison of the restricted air data with "good" data indicates that they are very similar, perhaps indistinguishable. Rows 11 and 12 (COMPRESSED TIME and CHOPPED) reveal misidentification between

compressed time data and chopped data as short find data. This may still be considered a success of the network: the data signatures were identified as "bad", and a clue is provided as to what is wrong with the data. In these cases, the FZ signal rose too soon, signifying a possible impact with something in the environment.

Table 29 shows similar results for the test data. All "good" data were classified as GOOD, and no "bad" entry was classified as GOOD with the exception of the restricted air test case.

While the results on this network were quite successful, this approach was abandoned due to the intensive design effort required for each network. To scale this network to the 6 channel, 8 operation problem posed by the material handling robotics application seemed unmanageable.

3.3 Statistical Processing

As a point of comparison, an alternative, statistically based approach was tested. Workcell problems are often accompanied by unusual forces, so we compared forces during operations to a statistical model of forces in "good" operations. Data from 60 "good" brushing operations were used to characterize the mean force and standard deviation for each 0.02 second segment of the process. The deviation of collected data for each time segment was calculated. If any channel deviated from the mean value by 4 times the standard deviation, a warning condition was set. If a warning lasted for at least 0.1 second, the operation was classified as "bad". The performance was not sensitive to small changes in either the deviation threshold value or the time threshold value.

This simple decision rule was applied to the 60 brushing operations used in building the statistics, to 10 "good" brushing operations not represented in the statistics, to 8 cases of restricted air flow (RES), and to 4 cases of tool not enabled (TNE). No attempt was made to classify the nature of the problem, if any, for the operation. The results of this processing are shown in Table 30.

	Performance of Statistical Processing on Collected Brush Data	
	Actual Classifications	
Desired Classi-fication	GOOD	BAD
GOOD	60 + 10	
TNE		4
RES	3	5

Table 30

All of the "good" operations were correctly classified. The TNE cases were correctly classified as "bad". Only a little over half of the RES cases were correctly classified as "bad"; this is consistent with the fact that the differences in the collected data are more marked when the tool is not enabled than when the air flow is merely restricted. These results are quite similar to those for the GRNN network on the same data. However the neural network has advantages in handling other types of failures and in providing a classification of the problem.

4. Discussion

The work described here demonstrates that the use of Artificial Neural Nets can be a viable technique for failure detection in robotic operations using currently available data. For both the material handling workcell and the deburring workcell, ANNs were able reliably to generate appropriate shutdown signals in response to error conditions while allowing normal operations to proceed. Although they were not error-free, the error rate was low enough to be acceptable in operation.

The computational workload varies with the implementation, but is moderate: up to 2.3 megaflops. This makes implementation of a real-time workcell monitor feasible.

However, in this application we found that significant network preprocessing and human judgment were required. When we attempted to just provide raw data and let the nets develop a classification scheme, the large volume of data and broad generalization required led to training problems.

Perhaps because of the preprocessing and chosen network configuration, the behavior of the successful networks seems to resemble techniques which may have a simpler implementation (e.g., envelope detection). While the neural nets successfully perform the monitoring task, they appear to have a disadvantage in training complexity without offsetting advantages.

The use of the Chen/Thomas/Nixon network has both advantages and disadvantages. The primary advantages are a straightforward training procedure and the fact that new data may be easily integrated into a previously trained network. One minor disadvantage is that "bad" data points among the "good" training data set are not naturally flagged by the training procedure, as they might be during training of a backpropagation net. Execution time and data storage will probably exceed that of a backpropagation network trained to do the same job. The curse of dimensionality affects training time, training set size, and ultimate network size for this type of network, although it affects most other types of network as well.

There is another type of neural network which also bears investigation for this type of application, also. That is the Adaptive Resonance Theory networks developed by Grossberg and Carpenter [Carpenter87 and Carpenter88]. These networks appear promising because they might be trained to deal with unfamiliar inputs correctly without generating the "bad" data grid that this work required.

Bibliography

Carpenter, Gail A., and Stephen Grossberg, "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns", *Applied Optics* pp. 4919-4930, 1987.

Carpenter, Gail A., and Stephen Grossberg, "The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network," *Computer*, March 1988 pp. 77-88.

Chen, Yan Qiu, David W. Thomas, and Mark S. Nixon, "Generating-Shrinking Algorithm for Learning Arbitrary Classification," *Neural Networks,* 1994, Volume 7, No. 9, pp. 1477-1489.

Moore, B., "ART1 and Pattern Clustering," *Proceedings of the 1988 Connectionist Models Summer School,* Pittsburgh, 1988, eds. D Touretsky, G. Hinton, and T. Sejnowski, pp. 174-185.

DISTRIBUTION:

1	MS-0188	LDRD Office, 4523
1	MS-1002	P. J. Eicker, 9600
1	MS-1006	P. Garcia, 9671
5	MS-1006	L.P. Ray, 9671
5	MS-0503	S. Tucker, 2338
6	MS-1008	S. Blauwkamp, 9621
1	MS-9018	Central Technical Files, 8940-2
5	MS-0899	Technical Library, 4414
2	MS-0619	Review & Approval Desk, 12690 For DOE/OSTI

SAND97-0957
Unlimited Release
Printed April 1997

Distribution
Category UC-706

Artificial Awareness for Robots

Using Artificial Neural Nets to Monitor Robotic Workcells

Susan D. Tucker
Control Subsystems Department

Lawrence P. Ray
Intelligent Systems Department II

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1006

Abstract

Current robotic systems are unable to recognize most unexpected changes in the work environment, such as tool breakage, workpiece motion, or sensor failure. Unless halted by a human operator, they are likely to continue actions that are at best inappropriate, and at worst may cause damage to the workpiece or robot. This project investigated use of Artificial Neural Networks (ANNs) to learn the expected characteristics of sensor data during normal operations, recognize when data no longer is consistent with normal operation, suspend operations and alert a human operator. Data on force and torque applied at the robot tool tip were collected from two workcells: a robotic deburring system and a robot material-handling system. Data were collected for normal operations and for operations in which a fault condition was introduced. Data simulating sensor failure and excessive sensor noise were generated. Artificial Neural Networks (ANN) were trained to classify operating conditions; several ANN architectures were tested. The selected ANNs were able to correctly classify all valid operating conditions and the majority of fault conditions over the entire range of operating conditions, having "learned" the expected force/torque data. Most faults introduced appreciable error in the data; these were correctly classified. However, a small minority of faults did not give rise to a detectable difference in force and torque data. It is believed that these faults could be detected using other sensors. The computational workload varies with the implementation, but is moderate: up to 2.3 megaflops. This makes implementation of a real-time workcell monitor feasible.

Intentionally Left Blank