

2003

# Finding Unexpected Events in Staring Continuous-Dwell Sensor Data Streams Via Adaptive Prediction

Peter G. Raeth

Nova Southeastern University, [peter\\_raeth@ameritech.net](mailto:peter_raeth@ameritech.net)

This document is a product of extensive research conducted at the Nova Southeastern University [College of Engineering and Computing](#). For more information on research and degree programs at the NSU College of Engineering and Computing, please click [here](#).

Follow this and additional works at: [http://nsuworks.nova.edu/gscis\\_etd](http://nsuworks.nova.edu/gscis_etd)

 Part of the [Computer Sciences Commons](#)

## Share Feedback About This Item

---

### NSUWorks Citation

Peter G. Raeth. 2003. *Finding Unexpected Events in Staring Continuous-Dwell Sensor Data Streams Via Adaptive Prediction*. Doctoral dissertation. Nova Southeastern University. Retrieved from NSUWorks, Graduate School of Computer and Information Sciences. (785)  
[http://nsuworks.nova.edu/gscis\\_etd/785](http://nsuworks.nova.edu/gscis_etd/785).

This Dissertation is brought to you by the College of Engineering and Computing at NSUWorks. It has been accepted for inclusion in CEC Theses and Dissertations by an authorized administrator of NSUWorks. For more information, please contact [nsuworks@nova.edu](mailto:nsuworks@nova.edu).

Finding Unexpected Events in Staring Continuous-Dwell  
Sensor Data Streams Via Adaptive Prediction

by

Peter G. Raeth

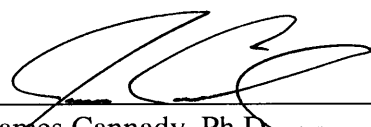
A dissertation submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy

Graduate School of Computer and Information Sciences  
Nova Southeastern University

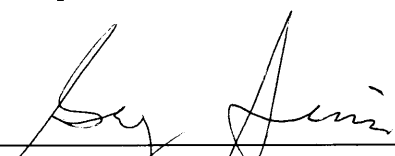
2003



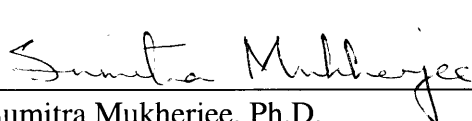
We hereby certify that this dissertation, submitted by Peter G. Raeth, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

  
\_\_\_\_\_  
James Cannady, Ph.D.  
Chairperson of the Dissertation Committee

3/6/03  
Date


  
\_\_\_\_\_  
Greg Simco, Ph.D.  
Dissertation Committee Member

3/6/03  
Date

  
\_\_\_\_\_  
Sumitra Mukherjee, Ph.D.  
Dissertation Committee Member

3/6/03  
Date

Approved:

  
\_\_\_\_\_  
Edward Lieblein, Ph.D.  
Dean, Graduate School of Computer and Information Sciences

3-6-03

Date

Graduate School of Computer and Information Sciences  
Nova Southeastern University  
2003

## **Dedication**

Without the blessings of God, this dissertation would not have been possible. The author is most thankful for His gifts, not the least of which are wife Marilyn and son Dan.

## In Memoriam

This dissertation started under the mentorship of Rollins Guild. His guidance to this student went far beyond the technical subjects of University study and had a deep personal affect. His setting an example of what a prime professional is at the doctoral level helped immensely in my formation. With his passing, we lost a leader whose influence goes much further than the courses and the path through the University.

In meditating on his continuing impact, I came upon something spoken a long time ago by a Native American. The author is lost to us now but his words speak to the present and future as much as they did to the people of that time.

*I give you this one thought to keep --*

*I am with you still -- I do not sleep.*

*I am a thousand winds that blow,*

*I am the diamond glints on snow,*

*I am the sunlight on ripened grain,*

*I am the gentle autumn rain.*

*When you awaken in the morning's hush,*

*I am the swift, uplifting rush of quiet birds in circled flight.*

*I am the soft stars that shine at night.*

*Do not think of me as gone --*

*I am with you still -- in each new dawn.*

## **Acknowledgements**

The author is deeply indebted to James Cannady, Greg Simco, and Sumitra Mukherjee. Transitioning from the Masters to the Ph.D. level is a huge step, much larger and difficult than those who have not taken it can imagine. The committee is essential to making the journey successfully since they guide you through the necessary personal transformation. They also help you lay the foundation for success in a new way of life and a new career.

Not to be forgotten is Junping Sun. He taught the author's first course at Nova and offered tremendous encouragement and guidance as this research took its first steps.

Don Bertke, Ball Aerospace & Technologies Corp, acted as the author's business mentor. All technology needs a meaningful purpose. Don provided ongoing discussion on possible uses of the technology developed by this research.

Allan Mord, Ball Aerospace & Technologies Corp, acted as the author's marketing mentor. Allan arranged access to a number of potential customers who provided data and insight into difficult problems needing solution.

Randy Bostick, National Air Intelligence Center, asked the original question that triggered this research. He asked a very precise, concise, and crisp question. The author would not be the first to remark on the value of having a clear problem statement. Randy also participated by providing data and discussion.

## **Abstract**

This research produced a Predictive Anomaly Detector (PAD). It is an adaptive prediction-based approach to detecting unexpected events in data streams drawn from staring continuous-dwell sensors. The underlying technology is spectrum independent and does not depend on correlated data (neither temporal nor spatial) to achieve improved detection and extraction in highly robust environments. ("robust environment" refers to the data stream's control law being variable and the spectral content covering a wide range of wavelengths.)

The resulting approach uses a network of simple building-block equations (basis functions) to predict the non-event data and thereby present subtle sub-streams to a detection model as potential events of interest. The prediction model is automatically created from sequential observations of the data stream. Once model construction is complete, it continues to evolve as new samples arrive. Each sample value that is sufficiently different from the model's predicted value is postulated as an unexpected event. A subsequent detection model uses a set of rules to confirm unexpected events while ignoring outliers.

Intruder detection in robust video scenes is the main focus, although one demonstration achieved voice detection in a noisy audio signal. These demonstrations are coupled to a concept of operations that emphasizes the spectrum-independence of this approach and its integration with other processing requirements such as target recognition and tracking.

Primary benefits delivered by this work include the ability to process large data volumes for obscured or buried information within highly active environments. The fully automated nature of this technique helps mitigate manning shortfalls typically associated with sorting through large volumes of surveillance data using trained analysts. This approach enables an organization to perform automated cueing for these analysts so that they spend less time examining data where nothing of interest exists. This maximizes the value of skilled personnel by using them to assess data with true potential. In this way, larger data volumes can be processed in a shorter period of time leading to a higher likelihood that important events and signals will be found, analyzed, and acted upon.

## Table of Contents

<b>Dedication</b>	<b>iii</b>
<b>In Memoriam</b>	<b>iv</b>
<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>List of Figures</b>	<b>xi</b>

### Chapters

#### **1. Introduction 1**

<i>Problem Statement</i>	<i>1</i>
<i>Goal</i>	<i>2</i>
<i>Relevance, Significance, and Need</i>	<i>5</i>
<i>Barriers and Issues</i>	<i>8</i>
Signal Processing	8
Image Streams	9
Issues	11
<i>Research Elements Investigated</i>	<i>12</i>
<i>Limitations of the Study</i>	<i>18</i>
<i>Definition of Terms</i>	<i>19</i>
<i>Summary</i>	<i>28</i>

#### **2. Review of the Literature 30**

<i>Overview</i>	<i>30</i>
<i>Historical Background</i>	<i>31</i>
Further Evolution	32
<i>Existence of the Problem</i>	<i>34</i>
General	34
Image Sequence Analysis	35
Intrusion Detection	36
Implementation	37
<i>Practical Applications of the Solution</i>	<i>38</i>
Business/Management	38
Technical Applications In Engineering	39
<i>Implementation Technology</i>	<i>40</i>
Neural Networks	40
Basis Function Networks	41
Time-Series Analysis	44
Tools	45
<i>Metrics</i>	<i>46</i>
<i>Reference Material</i>	<i>47</i>
Specialized Dictionaries	47
Data Extraction and Preparation	47
Supporting Information	49
Change Detection	50

### **3. Methodology 51**

#### *A Philosophy of Adaptive Automation Development 51*

Working Definition 51

Process Review 52

Know the End User's Goals and Have Access to Domain Knowledge 53

Choose the Development Task 54

Select an Appropriate Data Set 55

Choose the Algorithms and Heuristics 56

Clean the Data 56

Perform Appropriate Data Reduction and Transformation 57

Apply the Selected Algorithms and Heuristics to Accomplish the Task 57

Interpret and Validate the Results 58

Iterate Through the Above Steps as Needed 58

Consolidate, Review, and Document the Results 60

Install the Results in the Target Environment 60

Supporting Technologies 61

Summary 65

#### *Functional Experiments 66*

Experiments in Prediction 67

Comparing the Gaussian Predictor with a Linear Predictor 67

Single-Dimensioned Data Streams 71

Two-Dimensional Data Stream 74

Experiments with Detection 75

Detection in Single-Dimensioned Data Streams 78

Detection in Two-Dimensional Data Streams 81

Implementation Experiments 83

Single-Machine Single-CPU Throughput Experiment 86

Multi-CPU Throughput Experiments 86

Multi-Machine Multi-CPU Experiments 88

Experiment #1A 89

Experiment #1B 90

Experiment #1C 90

Experiment #2 90

Experiment #3 91

Exploring Issues In Moving Toward FPGA Implementation 92

#### **4. Results 94**

##### *Results From Applying the Theory 94*

###### Experiments in Prediction 94

Comparing the Gaussian Predictor with a Linear Predictor 94

Single-Dimensioned Data Streams 104

Two-Dimensional Data Stream 110

###### Experiments with Detection 111

Detection in Single-Dimensioned Data Streams 111

Detection in Two-Dimensional Data Streams 123

##### *Results from the Implementation Experiments 130*

###### Single-Machine Single-CPU Throughput Experiment 131

The Original Trial 131

Comparison Trial 1 132

Comparison Trial 2 134

Comparison Trial 3 134

##### *Multi-CPU Throughput Experiments 135*

Issues in Parallel Processing 135

Opportunities 138

Experiment with a Uni-Dimensional Data Stream 140

###### Multi-Machine / Multi-CPU Experiments 141

Experiment 1A 144

Experiment 1B 145

Experiments 1C and 2 145

Experiment 3 147

Discussion 148

###### Exploring Issues in Moving Toward FPGA Implementation 154

Background 155

Importance 157

Approach 159

A Simple Java/Forge Case 161

Translating PAD/C++ to PAD/Java 165

Issues Encountered While Feeding Java Code to Forge 165

Casting Floating Point Values as Integer Variables 166

Testing the MathFP Java Fixed-Point Library 169

Test of MathFP Iterations 169

Test of MathFP Computing  $\text{Exp}(x)$  170

Summary and Recommendations 172

#### **5. Conclusion, Implications, Recommendations, Summary 176**

*Summary 176*

*Conclusion 176*

*Implications 177*

*Recommendations 179*



## **Appendixes**

- A. Mathematics Behind the Anomaly Detection Model's Prediction Phase 182**
- B. Mechanization of Statistical Significance Calculations 187**
- C. Derivation of the Equation for Estimating  $f$  in Amdahl's Equation 189**
- D. Support Provided to this Research 190**
- E. Author Biography 193**
- F. Reference List 194**

## **List of Figures**

1. *Illustration of thresholds drawn from simple moving averages* 13
2. *Illustration of thresholds based on fixed values* 14
3. *Illustration of thresholds drawn from prediction error* 15
4. *Expected Performance Drops at First With Attempted Improvements* 60
5. *Adaptive Automation Technology Continuum* 62
6. *3-Dimensional View of Adaptive Automation Technologies* 64
7. *Prediction as a Prelude to Detection* 66
8. *Chaos original signal magnitude* 69
9. *Fan noise original signal magnitude* 70
10. *Illustration of Chaotic Data* 73
11. *Illustration of Fan-Noise Data* 73
12. *Fan noise with loud "Hello" mixed in* 79
13. *Spectral plot comparing fan noise and the word "Hello"* 79
14. *Throughput improvement alternatives* 85
15. *Computer physical arrangement for multi-machine/multi-CPU tests* 89
16. *Spectral plot of the chaotic data* 96
17. *Spectral plot of the fan noise data* 97
18. *Comparison of range-relative prediction error for chaotic data* 98
19. *Comparison of standard deviation for chaotic data experiment* 99
20. *Comparing range-relative prediction error for fan-noise mapped data* 100
21. *Comparing standard deviation for the fan-noise mapped data* 101
22. *Comparing range-relative prediction error for fan-noise unmapped data* 102
23. *Comparing standard deviation for fan-noise unmapped data* 103
24. *Average RRPE for two-dimensional prediction experiment* 110
25. *Standard Deviation of Average RRPE in two-dimensional prediction* 111
26. *Original noise/voice signal* 113
27. *Linear-filtered signal* 114
28. *Non-linear filtered signal* 114
29. *Fan noise with voice reduced to 5% of original volume (unfiltered signal)* 115
30. *Fan noise with voice reduced to 5% of original volume (filtered signal)* 116

31. *Constant with rectangular pulse (unfiltered signal) 118*
32. *Constant with rectangular pulse (filtered signal) 118*
33. *Chaos with rectangular pulse (unfiltered signal) 119*
34. *Chaos with rectangular pulse (filtered signal) 120*
35. *Sine with cosine (unfiltered signal) 121*
36. *Sine with cosine (filtered signal) 121*
37. *Sine with cosine RRPE 122*
38. *Sine with cosine RRPE standard deviation 123*
39. *Screen shot during chaotic eyeball experiment 124*
40. *False Alarms prior to intruder entering the chaotic scene 125*
41. *False alarms after intruder entered the chaotic scene 125*
42. *Portion of good detects during chaotic eyeball experiment 126*
43. *Unfiltered and filtered benign scene as model car moves in the scene 128*
44. *Unfiltered and Filtered robust scene as model plane moves in the scene 130*
45. *Model architecture: single-CPU / single-machine 132*
46. *General Execution Speed Expectations With Parallel Processing 136*
47. *Amdahl's Law Predicting Execution Speedup Via Parallel Processing 137*
48. *Model architecture: single-machine / multiple-CPU's 140*
49. *Results from experiment with loop-splitting in a 141*
50. *Architecture for applying the model to image stream pixels 143*
51. *Comparing Execution Times 151*
52. *Comparing Speed-Up 152*
53. *Comparing Efficiency 153*
54. *Comparing FPGAs to traditional processing approaches 159*
55. *Example of CORDIC coordinate system rotation (Knaust) 168*
56. *Sample architecture for PAD employment 177*
57. *Hybrid adaptive systems for real-time sensor signal utilization 178*
58. *Illustration of the Gaussian centers,  $\xi_i$ , and the distance between them,  $\Delta$ . 183*
59. *Original audio signal 184*
60. *Fourier Transform plot of original audio signal 185*
61. *Example calculation of statistical significance 188*

## Chapter One

### Introduction

#### Problem Statement

This research provides an advance in univariate anomaly detection within the field of staring continuous-dwell sensor data processing. Mathematically, this anomaly detection problem can be expressed in the following way:

Consider a univariate data stream:  $X = \{x_1, x_2, \dots, x_t\}$  such that the  $x_j$  are data samples received in time sequential order. Find those subsets of  $X$ ,  $S = \{s_1, s_2, \dots, s_z\}$  such that each  $s_i$  is a contiguous sequence within  $X$  that is not normally a part of  $X$  (an anomalous sequence) according to some criteria. This criteria is informed only by the samples received prior to the anomalous sequence  $s_i = \{x_b, x_{b+1}, \dots, x_{b+c}\}$  where  $c$  is the number of samples in the anomalous sequence.

Discussions with sensor analysts reveal that typical criteria for anomaly detection uses thresholds that are preset. However, the highly dynamic data streams delivered by typical sensors require a thresholding method that does not depend on presets but that adapts continuously. The basic issue with preset thresholds is that if the setting is too high then too much of the data stream is discarded. If the setting is too low, the event of interest is obscured. This research developed a solution to this problem for situations that do not involve white noise. The basic approach is to take advantage of data streams' temporal nature by adapting to their evolving control laws. By predicting the next data sample it is possible to eliminate non-static but expected content while retaining unexpected anomalies. In this way, it is possible to estimate the  $s_i$ . The method achieved here is judged by the number of  $x_j$  not found to be a part of some  $s_i$  when those  $x_j$  actually are anomalies. These are the missed detects. False alarms are those  $x_k$  thought to be a member of some  $s_i$  but which are not really anomalies. Synthetic data streams make it

easy to inject anomalies and to precisely determine missed detects and false alarms. In natural data streams anomalies can be some event of interest or they can be due to some momentary variation. In this case, the metrics need to be estimated.

Starting with the seminal work of Sanner and Slotine (1991, May/November) in control theory, this research developed a modeling technique that implements the required adaptive detection threshold. This model enables the detection of unspecified anomalies in spectrum-independent staring continuous-dwell sensor data streams. The resulting technique works with signals taken directly from the sensor with minimal preprocessing or filtering applied, other than converting voltage into engineering units. In essence, this work takes a technique from control theory and combines it with ideas from image and signal processing to extend the available technology for anomaly detection.

## **Goal**

Detection accuracy was the goal of this research. Since a typical data stream's control law is unknown and dynamic, it is difficult to know how to set a fixed detection threshold so that the expected content is filtered out while the anomalies are retained. Developed by this research was an approach that automatically builds a detection threshold that adapts to staring continuous-dwell sensor data streams by using numeric sequence prediction. While some primitive preconceived notions can be built into such a model, it is impossible to build a model that describes all possible data streams. To make this work, the model has to be built on the fly as the data stream evolves. A method was established for evolving the model as the data stream's control law evolves and to create a means for confirming the beginning and end of an anomaly. Demonstrations include data from chaotic equations, actual audio, and actual video. Measurements of accuracy include false alarms, missed detections, and good detections. The research sought to maximize the number of good detections while minimizing the number of false alarms and missed detections.

There are two ways to measure detection accuracy. For missed detects and good detects, one could measure the number of anomalies that were detected and compare that to the total number of anomalies. This basic measure does not count the number of samples the anomaly occupied but simply asks if any sample within an anomalous block was found. An extension to this idea is to require a certain number of contiguous samples to be found within an anomalous block before the anomaly can be said to have been found. False alarms can similarly be measured by the number of "anomalies" that get detected that are not true anomalies according to established ground truth. For these measures, a single anomaly would be a group of contiguous samples that are so tagged. Coupled with the number of contiguous sample blocks, it is necessary to compare the time spent in anomalies with the total time. A good definition of "contiguous" is also needed. How many samples does it take to separate one block of samples from another? This depends on the application. One approach is to directly relate the number of samples required to separate a block to how dynamic the data stream is.

Another approach to measuring detection accuracy is to first count the total number of samples that are actual anomalies. One would then compare that number to the number of samples with detected anomalies. This would give a count of good and missed detects. False alarms would be measured by the number of samples tagged as anomalies that were not. This number would be compared to the number of total samples minus the number of samples that contained true anomalies.

High or low detection accuracy is dependant on the application. For instance, if the anomaly is only present for ten samples and the detector uses four samples to verify detection then there will be a minimum of 40% missed-detects if the measure is by sample. However, if the measure is by the number of anomalies then the accuracy could be as high as 100% under the same circumstances. For false alarms, one has to pay attention to how many samples the detector needs to verify that an anomaly no longer exists. If the detector uses five samples to verify that an anomaly no longer exists then there will be at least five false alarms for every anomaly if the count is by samples. However, if counted on an event basis the false alarms contiguous to an anomaly would

not be counted. In either case, one would want the false alarms to be very low, around 10-20% in most cases. Otherwise, one runs the risk of replacing signal clutter with false alarm clutter. For detections, one would expect to find 80-100% of the anomalies.

Anomaly detection methods for continuous data streams need to deliver results soon enough to make a positive difference to the decision-making process impacting a rapidly evolving situation. Sampling rates and timeline requirements vary for each application but experience has shown that the typical desktop computer is not fast enough. Even for post-collection analysis, the volume of data is so high that a slow method is of little help.

Given the costs involved, this research did not achieve real-time processing. However, the research developed a detection model that is linearly scalable in the mathematical and computational sense. The demonstration of scalability used two networked quad-processor systems acting as a single virtual machine. Measurements included the clock time required to process each sample vs. the number of CPUs and the size of the model. Measures of data input time, data display time, and data storage time, were taken. Where multiple machines are involved, measures of inter-machine data transmission time were taken.

The research evolved the anomaly detection method using simulated data streams built from chaotic equations. This allowed the injection of anomalies at known points in the data stream, thus enabling precise measures of effectiveness. Once the method performed satisfactorily on simulated data streams, tests using natural data streams were performed (audio and video). These were less precise since it is hard to know the exact location of anomalies. However, it was possible to perform a reasonable estimate.

While this work dealt only with a workstation/network environment and did not attempt real-time processing or integration into an operational environment, a discussion on alternative approaches to throughput improvement is presented.

## Relevance, Significance, and Need

An ability to automatically find unspecified anomalies in staring continuous-dwell sensor data streams from staring continuous-dwell sensors is important to such endeavors as:

- surveillance, security, perimeter defense, treaty monitoring
- launch detection, signal filtering, signal detection
- environmental observation, process monitoring, malfunction detection
- health maintenance, weather prediction, natural resource management

The primary benefit is the reduction of large data volumes so that only the unexpected content remains while providing continuous remote monitoring. Other benefits accrue:

- Enhanced data stream filtering
  - ability to filter staring continuous-dwell sensor data streams for unspecified anomalies
- Labor-reducing process helps mitigate manning shortages
  - current methods typically depend on human observation and extraction
  - content not expected in the ongoing data stream can be automatically extracted and passed on to further processing or presented for human evaluation and response
  - direct human observation and evaluation of the entire data stream would no longer be necessary
- Enhanced analysis technology
  - the mathematics (Appendix A) is highly parallel with strong opportunities for real-time implementation
  - model adapts automatically as the data stream's control law evolves



Analyzing continuously-sampled data streams is an increasingly complex activity due to several major factors:

<ul style="list-style-type: none"> <li>• improvements in sensor sampling rates, resolution, and sensitivity</li> </ul>	<ul style="list-style-type: none"> <li>• expanding data volumes are creating a progressive analysis backlog</li> </ul>
<ul style="list-style-type: none"> <li>• the number and types of data streams to capture and analyze is growing</li> </ul>	<ul style="list-style-type: none"> <li>• new anomalies of interest are more difficult to detect and identify</li> </ul>

These factors make it is necessary to explore new ways to automate data stream analysis.

Generally, the literature suggests that anomaly detection methods typically make one or more of the following assumptions. An operational scenario may make these assumptions unacceptable.

<ul style="list-style-type: none"> <li>• benign or static backgrounds</li> </ul>	<ul style="list-style-type: none"> <li>• events of interest are known</li> </ul>
<ul style="list-style-type: none"> <li>• uniform backgrounds</li> </ul>	<ul style="list-style-type: none"> <li>• known spectra</li> </ul>
<ul style="list-style-type: none"> <li>• the control law generating the data does not change</li> </ul>	<ul style="list-style-type: none"> <li>• correlated data is available</li> </ul>
<ul style="list-style-type: none"> <li>• the data stream can be pre-characterized via some external modeling process using apriori knowledge</li> </ul>	<ul style="list-style-type: none"> <li>• cooperative data streams exist</li> </ul>

From this discussion, it is clear that a generic model that can automatically adjust itself to staring continuous-dwell sensor data streams and detect unspecified anomalies is needed. This project created a linearly scalable technology that accomplishes anomaly detection while minimizing common assumptions.

These next citations are further evidence of the relevance of this research. They are drawn from Frost and Sullivan's (1999) study of the remote sensing market. The author's comments are in parentheses.

pp. 2-8, 2-10, 2-11: The current demand for data products is restrained by the lack of availability. This is a major issue now and will be at least a medium issue for the next five years. Currently, there are 25-30 countries who use commercial and military remote sensing data. The demand within those nations already using such data will grow in proportion to the number of satellite data products available and the affordability that competition can bring. (Thus, there is an unrealized demand for data products and the related timely analyses.)

p 2-5: Worldwide demand for high-resolution data will remain high for at least the next five years. Over the next five years there will be a constantly growing commercialization of satellite data. (Coupled with their projection above, this is a confirmation of the unrealized demand for spectral data with the related increase in data flow.)

p 2-5: There is a strong probability that the demand for surveillance and change monitoring will be recession proof for the next five years. There is a demand for near real-time analysis results in regard to political situations, natural disasters, and treaty compliance. (As these situations become more complex and urgent, and as the required detections become more obscure, the more computationally burdensome will become the attendant analyses.)

p 2-6: There are about to be commercial launches of satellites that have a temporal resolution of 2-4 revisits per day. (There are also sensors that are continuous-dwell staring systems that provide a continuous data stream. In any case, the ability to respond to that much information puts us in a situation of having to upgrade our ability to make best use of existing computational infrastructures.)

pp. 2-6, 2-7: The amount of data commercial remote sensing satellites will bring back will be of great value to agriculture; oil, gas, and mineral exploration; environmental monitoring; and ocean monitoring to name a few. Commercial satellite data will increasingly provide critical information layers for specialized data products in a myriad of industries. The economies of scale several satellite operators will provide will continue downward pressure on data pricing, and should enable more purchases from existing customers and new purchases from the commercial world. A great majority, as high as 80%, of high-resolution commercially-obtained satellite data sales to foreign governments will be used for surveillance in a certain country's region of influence. The ability to observe one's borders and one's neighbors is highly coveted.

p 3-15: The ability to provide real-time analyses is a competitive factor. (Timely analyses becomes more difficult to achieve as the data rate increases. From their

projections above, there is a clear chance that the data rates will indeed increase and the urgency for which these analyses are required will also increase.)

p 2-12: For satellite data providers selling in their own countries, profitability will be aided by an ability to team with resellers or vertically integrate a value-added service to stay as close as possible to the end user. Value-added off-the-shelf data products will be more and more common across the remote-sensing marketplace.

p 1-6: PC computing power is a revolutionary end-user enabler. (It is important to make best use of existing resources to provide solutions that take advantage of a user's office or laboratory network of computational devices.)

p 1-4, 1-5: The world remote-sensing and geographic information systems (GIS) software markets stood at \$139 million in 1998. The compound annual growth rate (CAGR) for this market is forecast to be 17.1 percent through 2005. Remote sensing, in terms of revenues, will continue to be the dominant vehicle for remote-sensing and GIS data through 2005. Revenues for this market in 1998 were nearly \$2.2 billion, and the CAGR for this segment through 2005 is 9.6 percent.

## **Barriers and Issues**

The elusive part of this research is the detection of unspecified anomalies in evolving staring continuous-dwell sensor data streams. Building the model automatically (from scratch), evolving the model as the data stream's control law evolves, and confirming the beginning and end of anomalies is a non-trivial task. This is due to problem areas that bear on single-dimension and two-dimensional data streams.

## Signal Processing

Every signal contains a certain amount of noise. It is necessary to either account for the noise or the noise has to be filtered out in a pre-processing step. There are many means of filtering noise. These are described in such texts as Embree and Danieli (1999, pp. 177-330), Morgan (1999, pp. 201-262), as well as Otnes and Enochson (1978, pp. 106-218).

This research did not concentrate on applying filters per se but remained aware that the necessity may arise with certain datastreams.

Beyond signal noise, Otnes and Enochson (p 89) go on to list several other problem areas that can arise:

- computer word has too few bits to capture the necessary numeric accuracy
- extraneous electrical noise picked up from the environment
- noise added by amplifiers
- distortion caused by signal levels being too high
- excessive noise level owing to the signal level being too low
- frequency aliasing due to a too-low sampling rate
- aperture error in the sampling device
- jitter in the sampling device
- nonlinearities in the A/D converter
- dropouts in the converter
- converter noise
- operator error

Lockhart (1995) talks a bit more about sampling rate. He reminds us about the Nyquist rate requiring that a bandwidth-limited signal needs to be sampled at just slightly greater than twice the highest frequency. For example, if a signal is limited to a top-end of 300hz, the sampling rate would need to be just slightly higher than 600hz. To achieve bandwidth limits, he recommends applying an anti-aliasing filter to the pre-sampled analog signal, although this may not always be necessary.

### Image Streams

Image stream problems are partially discussed by Russ (1999, pp. 80-83, pp. 194-195). Given the wide range of image types and sources, he raises several issues that apply to images gathered for digital processing. One of these issues is that the same type of feature should look the same wherever it appears. This implies that the color and brightness values should be the same and that the illumination is stable for frames taken over time. Another issue is that there be local sensitivity to variations in the image. This implies that edges and boundaries are well defined. He also says that the optics must be free of dirt and other sensor-based obstructions lest false specimens be introduced into the image.

His discussion of methods to correct non-uniform illumination require the assumption that the illumination be constant for at least some period of time. His approach to generating a specimen-free background appears to assume a static scene in which the background does not change. Carefully setting up the imaging conditions is the foundation of his discussion for alleviating collection issues.

Control over imaging conditions is something difficult to achieve in natural outdoor settings.<sup>1</sup> So it is necessary to adapt to illumination that varies over time and to a small amount of non-random vibration (jitter) in the sensor platform. By implication, this adaptation needs also to deal with non-static backgrounds. Further, it is unlikely that features will look the same wherever they appear or that their boundaries will be well defined. The research addressed these issues for applications where it is important to detect the arrival of an anomaly in a continuous sequence of samples or image frames. Should the anomaly be already present, the method at least detects the anomaly when it changes state in spectra measured by the sensor. This is important in situations involving natural or man-made camouflage. Also, the method ignores obstructions local to the sensor since it is not always possible to gain physical access to the sensor. Lastly, the method does not require traditional image processing methods such as edge detection, methods specific to audio processing, or other assumptions on the spectra the data stream is drawn from.

---

<sup>1</sup> This does not include controlled settings such as in manufacturing or process plants. One has only to look at Vision Systems Magazine (<http://vsd.pennnet.com/home.cfm>) to see that setting up appropriate imaging conditions, including illumination, is a highly refined engineering practice. This dissertation worked with data streams captured from nature's environment under conditions that can not be controlled. Hence the term "natural" as opposed to "controlled" conditions.

Additional reasons for this project's goal not being met are presented by Schalkoff (1989, pp. 1-4). He begins by stating that image processing algorithms, with their associated hardware and software, are typically application dependent. Thus, their applicability to new problems and spectra is limited. Adaptation of image processing technology to specific applications is a continuing challenge. He goes further by saying that, especially for real-time applications, the computer processing required can be overwhelming.

This research overcame these problems by employing basis functions to build and evolve a frame prediction model (Appendix A). In this model each pixel is taken as an independent data stream. So, a frame prediction model is composed of numerous smaller independent models. Each basis function and sub-model is independent of all others and thus permits full use of parallel processing hardware to improve throughput. This enabled a demonstration showing the linear scalability of the technology. Additionally, this technology is spectrum independent since, from the point of view of the model, it is simply dealing with vectors of numbers with no units applied. Where the numbers come from is not an issue to the model itself, although pre- and post-processing steps may become necessary depending on the application at hand. In all, the technology is very generic and is able to generalize over multiple applications and spectra.

### Issues

As this is an emerging technology, the following risks should be considered whenever a new application is attempted:

- Does the data stream contain significant random components such that forward prediction of essential elements is not possible?
- Can the model's accuracy be maintained with natural data streams? At what point does noisy data become an issue?
- How much jitter is tolerable? How well does the model cope with missed samples?
- Is the anomaly present prior to model initialization?

- Does the data stream evolve slowly enough that achievable sampling rates are sufficient for an effective model to be built?
- Can the processing be performed fast enough to enable timely response?
- Do anomalies exist for such a long time that the underlying data stream evolves to the point that it is radically different at the end of the anomaly than at the start?

### **Research Elements Investigated**

Starting with the seminal work of Sanner and Slotine (1991, May/November) in control theory, combined with ideas from image analysis and signal processing, the author developed an adaptive detection threshold that finds unspecified anomalies in staring continuous-dwell sensor data streams. The basic technique uses each sample to predict the next. When the next sample does not match the prediction, the method postulates a detection event. Only loose or no assumptions are needed on the content, source, or spectrum of the data stream. Chaotic equations, audio, and video were used for testing since that form of data is most accessible to the author.

A common means employed commercially for anomaly detection is to set a threshold based on apriori knowledge of the data stream and the types of anomalies expected. There are two basic approaches for doing this. One approach measures the difference between the current sample and the simple moving average of some number of past samples. The other approach checks to see if the current sample value is greater or less than some fixed value. The moving average approach is illustrated in Figure 1. There are two moving averages, 3 point and 20 point, plotted against a chaotic equation. In such a dynamic environment it is clear that this approach will not suffice for anomaly detection. It is true that with much less dynamic problems this approach could work.

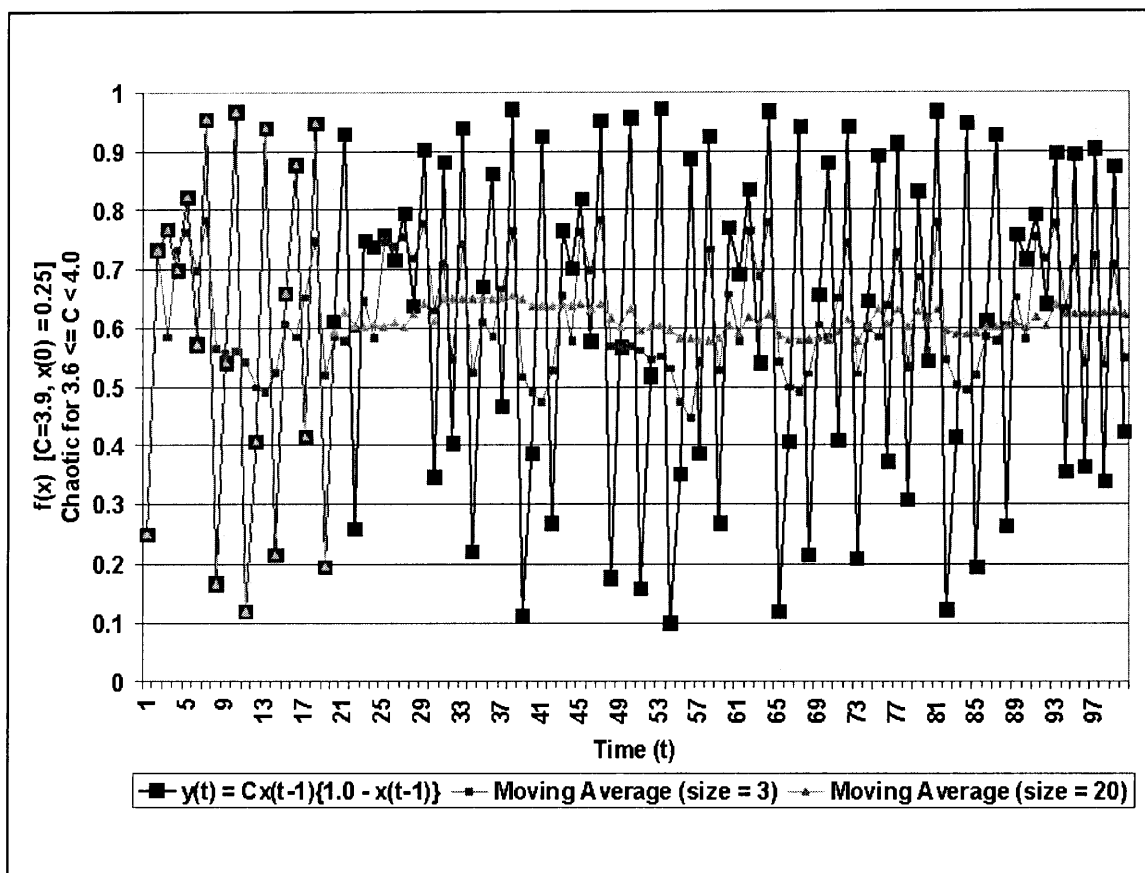
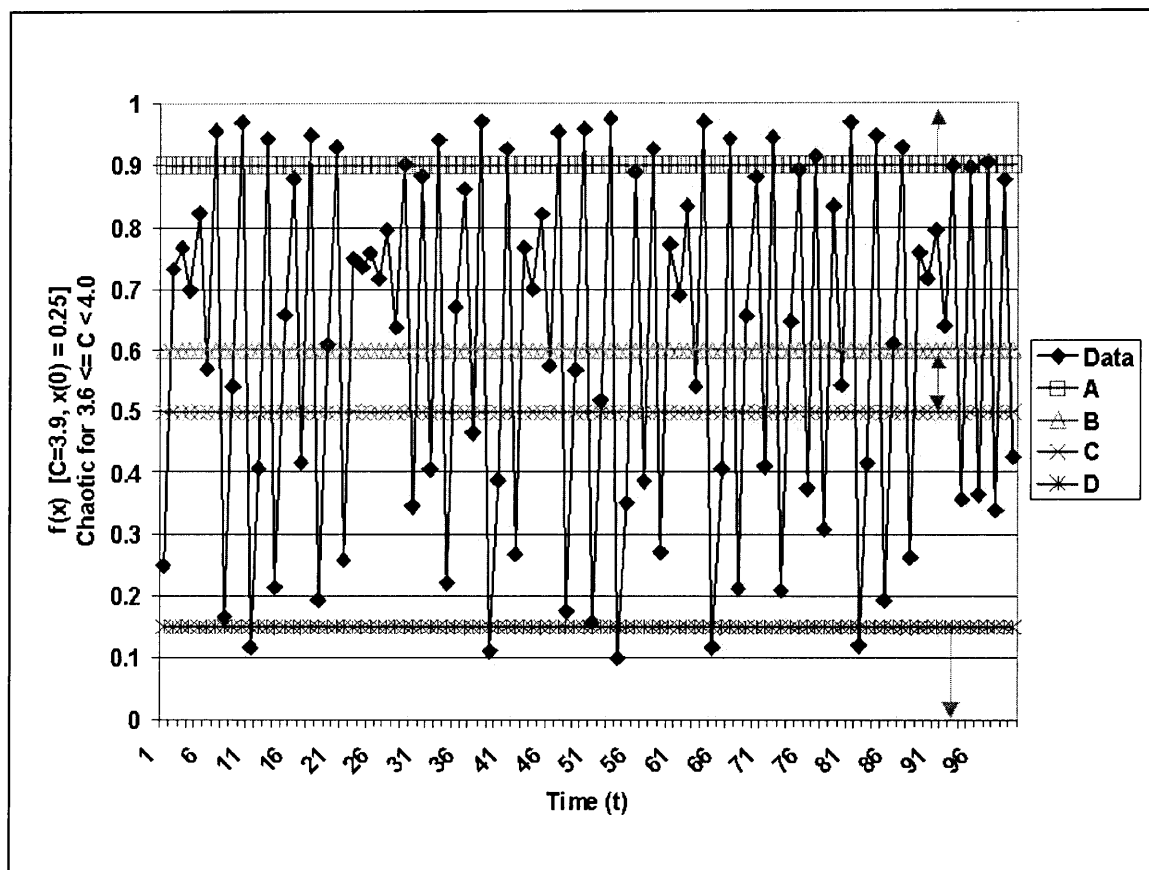


Figure 1. Illustration of thresholds drawn from simple moving averages

Figure 2 shows fixed-value thresholds. Anomalies are detected when sample values are greater than, less than, or in-between certain values. If these thresholds are set properly it is possible to catch anomalies that are outliers relative to the expected range of data. In this particular case, this approach to thresholds finds anomalies when sample data  $x$  is one of the following:

- $x > A$
- $x < D$
- $(x < B) \text{ AND } (x > C)$



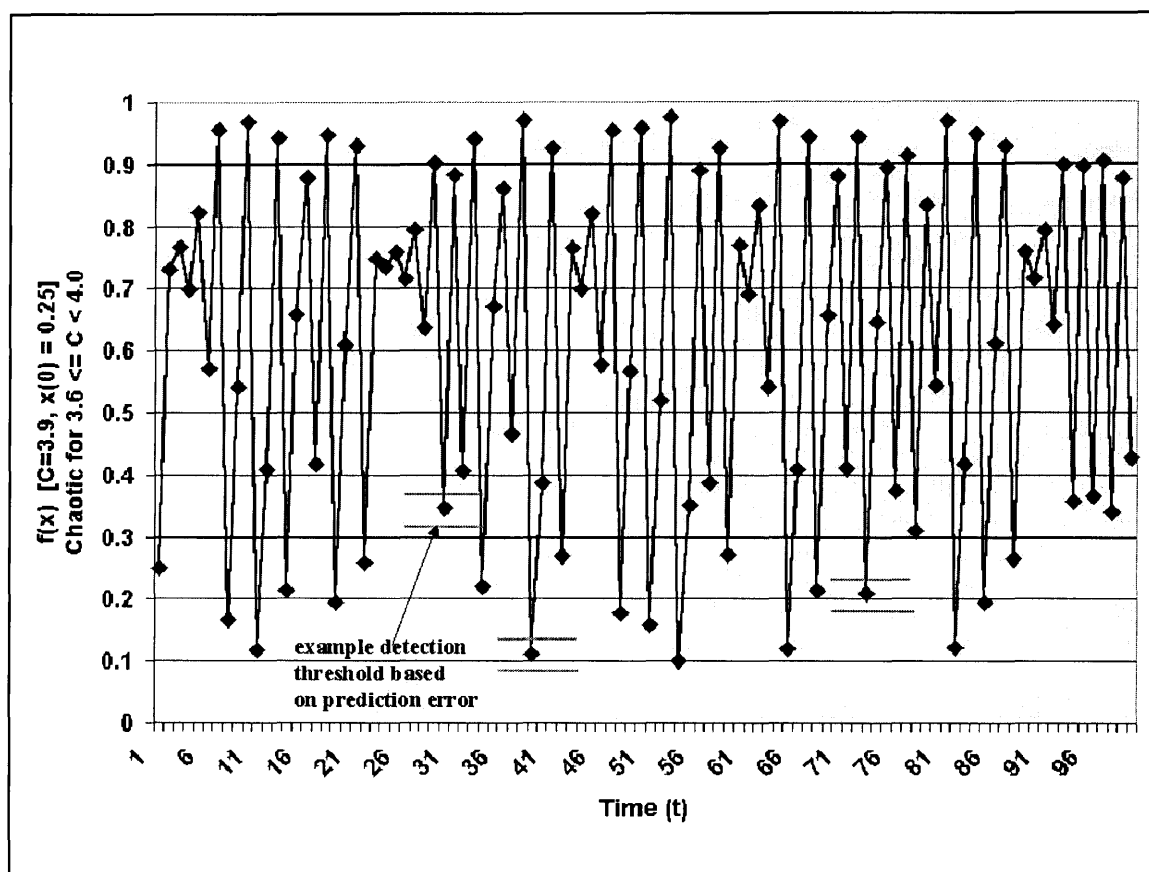


**Figure 2. Illustration of thresholds based on fixed values**

At issue with either of the above approaches is the heavy use of apriori knowledge concerning the data stream and the anomalies to expect. Further, they only catch range outliers. They do not catch anomalies relative to the control law generating the data. They also do not adapt well to evolving data streams. The current research minimized the need for apriori knowledge and created an ability to readily adapt to evolving control laws. In this researcher's experience, the more apriori knowledge used, the less flexible is the model when the data stream changes or some new anomaly occurs. It also becomes more difficult to modify the modeling approach for new requirements.

The threshold used here is based on the difference between the model's current-sample prediction error and the prediction error over some number of past samples (perhaps via the mean range-relative prediction error). Thus, in comparison to static sample-difference or moving-average thresholds, this new detection threshold floats more meaningfully

with the data. If the prediction error becomes too large, an anomaly is postulated. From this perspective, the level of noise in the data does matter but it is still possible to detect anomalies. This new threshold is illustrated in Figure 3.



**Figure 3. Illustration of thresholds drawn from prediction error**

It is true that, as currently employed, the prediction-error difference threshold itself is fixed. Prior experience shows that this threshold is not the same for all applications. Another impact comes from the level and length of noise bursts. Noise bursts can be taken as anomalies. If the control law of the noise does not suddenly change, the noise can usually be filtered out since nature is inherently non-random. Be that as it may, the detection threshold itself does vary with the model's ability predict the data stream. The currently-invariant aspect has to do with the allowed deviation from that threshold. The detection threshold itself changes with prediction ability and is based on the prediction error so that the detection threshold increases as does the prediction error.

Not only is nature inherently non-random, it is also non-linear. That is the reason this research uses a Gaussian model instead of the more popular linear models. This choice is backed up by the new IEEE802.3ab standard for Gigabit Ethernet. This standard calls for non-linear adaptive digital-echo and near-end crosstalk cancellation technologies because ever increasing data rates are causing difficult problems in those areas. Adaptive linear models have a very successful history and still have value. However, this research aimed at highly dynamic applications in which linear models have their limitations. Crosstalk is a similarly difficult dynamic problem that increasingly does not yield to filters based on linear models.

From the above discussion, this research adds to the field by:

- creating a non-static detection threshold that evolves with the data stream rather than use a static sample-difference or moving average threshold
- increasing the generality and flexibility of detection by going beyond current sample-difference measures
- creating a means of anomaly detection that is, with few exceptions, independent of data stream content, spectrum, and source
- providing a type of moving-target indicator that does not depend on the shape of the object or the clearness of its edges when the data stream is composed of image frames
- avoiding common assumptions:
  - benign or static backgrounds
  - uniform backgrounds
  - the data's control law does not change
  - known events of interest
  - known spectra
  - other domain data is correlated or cooperative

The adaptive modeling method employed here is composed of a network of independent Gaussian radial basis functions of a form similar to that used for classification (Raeth, Gustafson, and Little (1994)) although the constructive technique is much different and

the outputs are interpreted for analog rather than binary application. Each data stream has its own model. Each model is built automatically from sequential observations of the incoming samples, constituting a learning phase. As the model converges to the training input, it gradually becomes able to predict the next sample in the sequence. Once the model is complete, it continues to refine its predictions as new samples arrive. The model evolves by varying the heights of fixed-width Gaussians. The adaptation rate depends on the size of the model and the Fourier Transform frequency content of the data. One would expect the model to be fully trained and ready to enter operational mode within 60 data seconds. However, much depends on how long the signal's control law takes to generate sufficient output so that the signal can be fully characterized. (See the Issues section of this chapter.)

When an anomaly is detected, the model stops evolving until the anomaly is no longer present. In this way, the anomaly does not become part of the model's prediction baseline. This secondary control layer allows the anomaly's presence to be tracked and thus provides a type of moving target indicator as the anomaly undergoes temporal and spatial shifts. Anomalies are detected by the model using three measures of departure from the expected background:

- the number of sequential times the departure appears
- the number of sequential times the departure subsequently disappears
- the percentage difference between the actual and expected sample

Note that too sudden a change in the data stream's control law would be taken as an anomaly. In such a case, the prediction error would never fall below the detection threshold and the detector would continue to insist that an anomaly is present. A return to training mode would have to be triggered by some external event such as operator interaction. This is a case where correlated information would be needed. Something would have to indicate that the "anomaly" is really part of the expected data stream.

Basis functions are simple-equation building blocks that are a proven means of modeling more complex functions. Brown (writing in the book edited by Light, 1992, pp. 203-206) showed that if  $D$  is a compact subset of the  $k$ -dimensional region  $R^k$  then every

continuous real-valued function on  $D$  can be uniformly approximated by linear combinations of radial basis functions with centers in  $D$ . Proofs of this type have also been shown by Funahashi (1989); Girosi and Poggio (1989); and Hornik, Stinchcombe, and White (1989). They established the basic theory behind the predictive model produced by Sanner and Slotine (1991, May/November; 1991, December). Since the theory has already been well established, this research concentrated on a transition to the reality of anomaly detection in data streams, thereby creating a new technology and applying the theory to a new domain. This domain has come about because of the growing use of sensor data for such applications as security, defense, environment, and health. This research uses  $k = 1$ , a one-dimensional value such that  $R^k$  is a line. Look to Sanner and Slotine (1991, May/November; 1991, December) for an expansion to larger dimensions. A summary of the math for  $k = 1$  was published as part of a paper by Raeth, Bostick, and Bertke (1999). See Raeth and Bertke for additional material (2000). Appendix A contains a discussion of the math itself. This approach makes strong use of a data stream's temporal component and only makes the assumption that the data is not random, which is true, for instance, in the case of natural noise phenomena. Note though that the math assumes that the important elements of the data stream are spectrally limited in the Fourier Transform sense. (A low-pass filter could be configured to preserve the important elements required for sufficiently-accurate next-sample prediction.)

### **Limitations of the Study**

A primary limitation of this research was that it did not seek to achieve real-time throughput. However, there were some experiments in that arena that sought to make best use of existing networked computers. There was also a brief study on transitioning to Field-Programmable Gate Arrays. The intention here was to show that the required detection accuracy can be achieved in a reasonable amount of time with the potential to achieve true real-time processing.

All data sources were taken either from equations or post-collection from microphone or video sources. There was no attempt to read the sensors directly and process the data as it arrived. Rather, the data was recorded on disk for later processing.

The following assumptions were applied to the data stream. There are approaches to mitigate these assumptions but their exploration is outside the scope of this research project.

- there is a maximum spectral content in the Fourier Transform sense
- minimum and maximum values are known
- there is no missing data
- a staring continuous-dwell sensor is the source
- analog-to-digital conversion is already accomplished
- there is not a significant change in the data stream's control law while an event is in progress
- the data stream is not purely random
- signal sampling is at least at the Nyquist rate

### Definition of Terms

This index identifies terms appropriate to this line of study. Some of these are expanded from various authors as marked. Unmarked terms are from the present author.

**activity** Specific to a particular neural network node, the weighted sum of inputs to that node. For example, the activity of hidden node 1 would be  $INPUT1 * W(1,1) + INPUT2 * W(2,1) + \dots$ , where  $W(i,j)$  designates the weight on the signal coming from the  $i$ th input node and going to the  $j$ th hidden node. (Bungay, 1999)

**activation function** Specific to a particular neural network node. This is a basis function that receives the inputs to the node and produces the node's output. This function is typically non-linear and continuous.

**algorithms** Well-defined unambiguous procedures for performing operations that lead to exact solutions in an efficient manner. Although adaptive systems are more closely associated with heuristics, algorithms can be used for preprocessing and to acquire insights into the data. (Morris, 1992)

**anomaly** Data that is unexpected according to some criteria. Can also refer to data that is not similar to its neighbors in the temporal or spatial sense.

**anti-aliasing** A filter that attenuates frequencies in an analog signal that are greater than one-half the digital sampling frequency.

**basis function** Small building-block equation that can be combined with others to model more complex functions. An example is  $e^{-|x-y|}$ . With fixed  $y$  and varying values for  $x$ , these can be summed. Each  $x_i$  gives a local contribution to the overall model. Another example is the Fourier transform that uses combinations of  $\sin(x)$  and  $\cos(x)$  to build a model of a complex waveform.

**central maximum** Mid-point of a two-dimensional graph's independent axis

**continuous-dwell** While a sensor may gather data from the same "source", it may switch or "time share" between sub-sources. That is, for instance, it may always gather heat readings from System-A but individual readings may come from several measurement points on System-A. This refers to a staring sensor (see below) that is not continuous-dwell. A continuous-dwell sensor always gathers its data from the same measurement source. For a geographical sensor, a staring sensor may look at the same geographical area but will "sweep" the area. Continuous-dwell means that it does not sweep but just looks at the same place all the time.

**control law** The process that generates the analog signal from which a data stream is drawn. A function representing (modeling) the system that causes the observing sensor to generate a given data stream.

**data stream** An analog signal is converted to digital representation at a given rate. The periodic digital value returned by the analog-to-digital conversion process is called a data sample. A contiguous (in the temporal sense) set of samples is called a data stream.

**error** Difference between the computed output (using weighted basis functions) and the actual (correct) output. In training, the error is supposed to decrease to some specified minimum. (Bungay, 1999)

**evolutionary computing** A class of heuristics that attempt to solve a problem by means loosely based on natural evolution. A set of candidate solutions is evaluated in terms of solution quality, the better candidates are allowed to reproduce, sometimes with mutations. If one candidate comes to dominate the population, it is selected as the best solution. At times, different candidates are combined to make a new candidate. (Morris, 1992)

**expert systems** Applications in which problems are solved by means of an information base containing rules and data from which inferences are drawn on the basis of human experience and previously encountered situations. True enough, a task adaptive systems are often put to is the derivation of rules from data. However, there is no reason that previously established rules can not be used to guide a new analysis session. (Morris, 1992)

**field of view** Domain that is observable by the sensor. For instance, a camera might have a field of view of 45 degrees. This means that, without moving, it can observe a cone formed by a 45 degree angle relative to a line starting at the lens center and running at a zero degree angle with the pointing direction of the lens. Of course, there are range limits as well.



**frame** One sample from an image stream. An image stream is composed of a series of frames taken in time sequence. Note that "images" are not necessarily drawn from the visible spectrum.

**frame rate** Number of image frames acquired per time period.

**fuzzy logic** A method of machine reasoning that can process uncertain or incomplete information. This system of logic is based on the fuzzy set, a generalization of traditional two-state set concepts. Partial membership in a given set can take on the entire analog range of 0 .. 1 inclusive, rather than just the binary discrete values of 0 (not at all a member of the given set) and 1 (entirely a member of the given set and of no other). Fuzzy set theory applies these concepts to expert systems and estimates the degree of a conclusion's certainty. (Morris, 1992)

**gaussian radial basis function** One of many activation functions that can be used by a neural net node. This function can be specified and employed in many ways. This project's method uses the following form. The details of this form are explained in Appendix A.

$$g_i(x, \xi_i) = e^{-\pi \sigma_i^2 \|x - \xi_i\|^2}$$

**graph theory** The study of graphs, trees, and networks. A graph is a diagram or curve representing varying relationships between sets of data. "Data structures" is another topical area in this same genre. It is an organization of data in some characteristic way, such as arrays, trees, lists, queues, networks, or stacks. (Morris, 1992)

**grid** Set of points in two dimensions. All points are evenly spaced.

**ground truth** That which is known apriori about anomalies' temporal and spatial locations but not revealed to the detection model.

**hidden layer** Set of neural network nodes between the input layer and the output layer. Each node in the hidden layer "communicates" with each input node and with each output node. There may be more than one hidden layer. (Bungay, 1999)

**image** A grid of data streams that are organized according to their spatial relationships. It is not necessary that the data streams be from the visible spectrum nor that they all be drawn from the same spectral region. Also see frame.

**image size** Number of pixels (grid points) per frame.

**image stream** Series of frames taken in time sequence.

**input layer** Set of neural network nodes that receives and processes the input to a neural net. Can also be more generally seen as the pre-processing functions applied to the input data stream. (Terry, 1999)

**jitter** A small amount of non-random vibration or periodic movement in the sensor platform such that there is a small variation in the registration of the frame. Any periodic and ongoing lack of sensor-pointing instability.

**kurtosis** Degree of peakedness of a curve relative to a bell-shaped curve (other comparisons are possible). The larger the kurtosis, the more peaked the curve. (Spiegel, 1961, p91)

**mean square error** Takes the average of the sum of the squares of the difference between one value and another. Typically measures the difference between the actual and expected values. This method emphasizes large errors over small ones but is not weighted according to the type of error. (Masters, 1993, p344-346)

**natural data stream** Data streams captured from nature's environment under conditions that can not be controlled.

**neural networks** A computational network composed of mathematically defined nodes that are loosely based on biological neurons. These networks are often composed of one layer that receives, organizes, and processes the k-dimensional data input; a subsequent data processing (hidden) layer; and an output layer in which individual neurons identify particular patterns or deliver an analog result as an estimate of the system represented by the data. There are numerous neural network types. (Morris, 1992)

**neuron (Node)** Building block of a neural net. It is very loosely based on the brain's nerve cell. Neurons receive inputs via weighted links from other neurons. These inputs are processed according to the neurons' activation functions. Signals are then passed on to other neurons. There are 3 different types of neurons within a neural net. Input neurons receive, organize, and process encoded information from the external environment. Output neurons send signals out to the external environment in the form of an encoded answer to the problem represented by the input. Hidden neurons allow intermediate calculation between inputs and outputs. (Toulson and Toulson, 1999)

**OLAP** On-Line Analytical Processing. A category of database software that provides an interface that enables users to transform or limit raw data according to user-defined or pre-defined criteria, and to quickly and interactively examine the results in various dimensions of the data. OLAP primarily involves aggregating large amounts of diverse data. It can involve millions of data items with complex relationships. Its objective is to analyze these relationships and look for patterns, trends, and exceptions. (Dictionary.com, 2002)

**operations research** The mathematical analysis of optimization problems, especially of dynamic systems subject to a given set of constraints. The purpose of such analyses is to evaluate the effectiveness of a given system in achieving identified goals. (Morris, 1992)

**output layer** A set of neural network nodes that receives, organizes, and processes the output from the last hidden layer to produce a neural net's results. Neural network layers are numbered sequentially starting at the input layer. (Terry, 1999)

**pixel** One point on a frame's (image's) grid.

**process** Continuous or periodic series of actions organized and conducted to achieve an end result, such as a distillation process. May employ a system to facilitate the process. (Morris, 1992)

**real-time** Fast enough to continuously deliver results in time to have a positive impact on the decision-making processes impacting a rapidly evolving situation. A numeric specification is application dependant. In radar signal identification problems this may mean nano-second response times. In weather monitoring applications this may mean minutes or hours response time.

**registration** In regard to frames of data, the source referred to by a given cell. For perfect frame registration, the same cell will always refer to the same source. For spectral sensors taking data from a geographical location, the data for a given pixel should always be drawn from the exact same place.

**registration noise** Refers to the lack of perfect registration. For example, registration noise of one pixel means that the data stream for a given frame is being produced from a one-pixel variation relative to a center pixel. Such noise can be caused by sensor jitter but there are other causes such as recording inaccuracies.

**sample** Representative segment of a larger whole that is studied in order to gain information about the characteristics of the whole. A subset of an entire population that is observed in order to infer properties of the population. (Morris, 1992)

**sensor** A device that measures a system, environment, or other variable external to itself. A sensor can operate in any of a number of wavelengths and is not necessarily limited to those detectable by the six human senses. In the case of this research, it is necessary that the sensor provide a data stream composed of regular-interval samples.

**scene** Same as a frame or image stream, although meant in a sense larger than just the data. Speaks to what the frames are meant to measure or represent. For instance, the scene might be of the area beyond a security perimeter that is within the field of view of the sensor. The scene may be non-static in that it might contain, for instance, the movement of trees, the change in natural illumination as the sun runs its course, and the entry and exit of potential intruders.

**skewness** A data curve's property that describes its general shape. (Spiegel, 1961, p31) There are six general shapes that that curves can achieve in practice (viewed as the reader sees the page):

1. symmetrical, normal, or bell – data is equally distributed on either side of the central maximum, the curves skew would be 0
2. left skew – data is more heavily distributed to the left of the central maximum, the more positive the skew, the more heavily to the left the curve leans
3. right skew – data is more heavily distributed to the right of the central maximum, the more negative the skew, the more heavily to the right the curve leans
4. J – curve's maximum value occurs at the extreme right
5. Reverse J – curve's maximum value occurs at the extreme left
6. U – curve's maximum value occurs at both the extreme right and left
7. bimodal – curve has two maxima
8. multimodal – curve has multiple maxima

**spectra**, spectrum (singular) Set of wavelength limits within which a sensor suite can perform measurements.

**standard deviation** Measure of dispersion about the mean. The larger the value, the larger the dispersion. Tells whether the variable's values are grouped about the mean or not. (Bruning and Kintz, 1968, pp. 4-6; See Spiegel, 1961, p70, for calculation.)

**staring** The sensor gathers data from the same source.

**statistics** Mathematical methods for estimating, confirming, or denying population parameters from samples of varying degrees of accuracy. Adaptive systems do go beyond statistics. But, statistics can be a worthwhile start for determining basic characteristics of the data. (Morris, 1992)

**system** Combination of components, elements, sub-systems, and operating procedures that function together to achieve some purpose. May be used to facilitate a process. (Morris, 1992)

**training** The learning process during which a neural network develops the weighting coefficients that modify the nodes' inputs. Starting from an arbitrary initial set of weighting coefficients, the signals are input, weighted, summed, and output. Then the errors are computed and used to develop new weights. Iteration, error checking, and testing for convergence continue until the error is beneath a specified limit, at which point the network is said to be trained. (Bungay, 1999)

**t-test** (for difference between two independent means, assuming equal variances)  
Statistical judgment as to whether the performance difference between two groups is significant. Assumes already-constituted groups and that the experimenter wants to determine whether they differ with respect to some variable. Note that this is not the same as the Student t-Test. Note also that a simple variation of this test assumes unequal

variances. The most common use assumes equal variances. (Bruning and Kintz, 1968, pp. 9-12)

**variance** Measure of dispersion around the mean, given by the arithmetic average of squared deviations; the square of the standard deviation. Also known as the mean square deviation. (Morris, 1992)

**weights** Coefficients that are multiplicative factors of the inputs to each node. These weighted inputs are summed to produce the activity of a node, which is then used to determine the node's output. (Bungay, 1999)

## Summary

The technology established here could be termed a "Predictive Anomaly Detector (PAD)". Functionally, it acts as a dynamic squelch, minimizing the non-event background so that anomalous events (relative to the ambient background) can be more readily detected. This technology is pervasive, impacting many domains of application. It is not spectrum dependent nor does it rely on a deep knowledge of the domain's physics, sources, collection methods, or phenomenology. It is point-specific in that, even in the case of a spectral frame, it relies only on one-dimensional data streams (individual pixels). There is no need for cross-correlation with nearby pixels or related data streams.

Since the resulting detection model is so general, it tends to be fairly large. Therefore, this research also addressed the practical implementation of the model, although there was no attempt at true real-time processing. To facilitate the research, some simplifying assumptions were made so as to bound the project. These assumptions do not limit the practical application of the work nor do they draw results from toy problems.

Behind the detection model is a predictive method successfully used in control theory to find out-of-range machine operations so that appropriate feedback signals can be

generated. This method has already been documented in Sanner's publications. Transitioning this control theory method to applications in spectral analysis is not obvious and made for a non-trivial, original research project.



## Chapter Two

### Review of the Literature

#### Overview

This literature review highlights the documents from which the research drew its background. No one paper gives information specific to the main thrust. However, each paper contributes an essential element, the synthesis of which enabled progress. At the top level, this research leans heavily on Sanner's work in adaptive control theory. Basis functions form the mathematical foundation of his work and the anomaly detection approach described in this dissertation. Brown (writing in the book edited by Light, 1992, pp. 203-206) proved that if  $D$  is a compact subset of the  $k$ -dimensional region  $R^k$  then every continuous real-valued function on  $D$  can be approximated uniformly by linear combinations of radial basis functions with centers in  $D$ .

There are seven sections in this review. Each section groups papers according to a major topic. This derives from the fact that the present research requires the fusion of results provided by numerous people in many areas of investigation. The seven sections are:

- Historical Background
- Existence of the Problem
- Difficulty of Solution
- Practical Applications of the Solution
- Implementation Technology
- Metrics
- Reference Material

## Historical Background

Upward evolution of computer hardware speed, storage capacity, and data communications, along with advances in algorithms, heuristics, and programming theory, has enabled continual advancement in the ways computers can be employed. Taking advantage of this trend, people always want to solve new types of problems. These problems are ever more unstructured, poorly defined, and involving extreme volumes of sometimes incorrect, uncertain, and noisy data.

1954 saw the first commercial installation of a computer. Payroll processing was the application, a fairly straight-forward, well structured, and routine accounting procedure (Davis, 1974, p 3). Tasks such as payroll, test grading, record keeping, and data aggregation provide support for activities where there is great concern over the effective performance of specific, highly structured, well understood routine tasks. (See Thierauf, 1982, pp. 7-10, pp. 66-73) and Sprague and Carlson (1982, pp. 4-10, pp. 94-96) for a good discussion.) These were the advantages sought through the 1960s (Hussain and Hussain, 1981, pp. 5-6).

After solidifying the success for these types of tasks, the evolution continued by moving on to problems that are much less structured, embed several well-structured tasks, and seek to optimize the acquisition and use of resources in a search for optimal functionality. This is where Management Information Systems (MIS) comes into its full application. According to Sprague and Carlson (1982, p 7), the basic characteristics of MIS include structured information flows, a focus on single functional areas (production, marketing, personnel, etc), and report generation. MIS tends to be batch oriented with predefined inputs and outputs, and is not very interactive as we understand the term today.

Evolution in computer capability continued with Decision Support Systems (DSS). Here the computer is asked to help solve planning and other problems that are highly unstructured, poorly defined, and involving information that is less than precise. Rather than being task focused and batch oriented, DSS tend to be highly interactive, flexible,

adaptable, quick to respond, and supportive of needs that are not predefined (Sprague and Carlson, 1982, p 7).

### Further Evolution

The evolution in the ways people want to employ computers continues apace. Where straight forward applications like payroll processing were the original topic of pursuit, people now want to do things like build real-time systems that adapt to the problem at hand as that problem evolves. Where data volume used to be measured by the number of 96-byte Hollerith cards one had room for in a storage facility, we now have applications like NASA's EOS satellite constellation that is projected to transmit, store, and (hopefully) analyze 50 gigabytes of data every hour, all day, every day of the year (Way and Smith, 1991). Where the data was once fairly exact and the related models well established, we now deal in domains where the data is incorrect, uncertain, and noisy; the models can not be established by traditional means; and the associated heuristics are expected to adapt continuously to highly dynamic environments. From straight forward to highly complex, from well structured to unstructured, from well defined to poorly defined, from narrow application to broad, from static to dynamic, from clear data to uncertain, incorrect, and noisy: the trend continues through the present day and beyond.

This situation requires the next step in the evolution of the computer as a tool, a step that leads to adaptive systems. While the human-centered aspects of modern automation must be retained if the computer is to be useful, something more is needed. What is needed to solve modern problems is "... a new generation of techniques and tools with the ability to intelligently and automatically assist humans in analyzing the mountains of data ..." (Fayyad, Piatetsky-Shapiro, and Smyth, 1996, p 2). These new tools have to go beyond what has been done before. They must transcend even complex database queries and statistical analyses. Both of these only confirm or deny preconceived notions by analyzing the available data. They require a domain expert's prior insight into the data and the process being measured. While the role of the domain expert is still paramount,

these new methods need to be able to deliver results that do not involve preconceived hypotheses.

In treading upon this new ground, it is important to know where we want to go. Having stored "mountains of data", we want to achieve information and knowledge. It is essential to not confuse the two. The MIS and DSS literature tends to equate data with information and rarely mentions knowledge. However, they are not the same thing. Data is the raw material. Information is the result of data being processed in such a way that it is meaningful to the user in the context of some application. Knowledge is a truth or set of truths that span individual applications. So, data leads to information and information leads to knowledge. This present research concentrates on a new method for extracting useful information from large volumes of time-stamped data.

For extracting meaningful information, Fayyad, Piatetsky-Shapiro, and Smyth (1996, p 27) see changing data and evolving domains as one of the prime challenges. Other significant challenges these authors cite are:

- dealing with large databases
- high dimensionality
- overfitting
- assessing statistical significance
- missing and noisy data
- complex relationships between fields
- understandability of results
- user interaction
- use of prior knowledge
- integration with other systems

The solutions they briefly propose tend to deal with these issues at the post-collection level. However, it is this author's contention that methods performed solely post-collection can not have a real-time impact on highly dynamic processes. For a real-time impact to be made, analyses must be performed as the data is being collected. Adaptive models that are capable of this could also have value in post-collection analyses of large volumes of information drawn over time. Thus, there are two areas of potential impact:

real-time during collection such that the process can be modified immediately, and post-collection such that the process can be improved on a more fundamental scale.

Research and development experts can offer a valuable contribution to the solution of the challenges posed by rapidly evolving processes. This particular research deals with unlimited-length data streams where the model needed to represent that data stream is constantly changing. The desire is to modify the model fast enough to facilitate taking action that meaningfully impacts the process being measured. This is something already done in business, for instance, by stock traders who use computer programs to model the market in real-time and to automatically buy and sell (Goldman, 1988). In engineering, a common application is adaptive system control (Slotine and Li, 1991) whereby a model of the system is learned automatically and evolved continuously.

## **Existence of the Problem**

### General

According to Kerestecioglu (1993), the problem of detecting changes in dynamic systems is gaining in importance. His thrust pertains to the need for determining the existence of malfunctions and performance degradations in complex automatic systems. A similar train of thought is espoused by Sanner and Slotine (1991, May/November; 1991, December) as well as Akin and Sanner (1990). Their basic idea is to model the system in some way that undesirable changes in operation can be detected and acted upon. In the case of this research, "undesirable changes" are the anomalies that the model is supposed to detect.

This line of thinking is seconded by Morgan (1994, p 335). He says that one application of adaptive filters is the modeling of unknown systems. In the application pursued by this project, the technology is attempting to model the control law that is generating the data sequence, while having no knowledge of the source, so that unspecified anomalies in the sequence can be detected. Although Morgan's (1994) thrust is at unknown systems, the

author's experience has been that some known systems are so complicated that traditional modeling techniques can prove to be intractable, even if it is possible to gather full technical information. Thus, adaptive methods that learn from example have a place in modeling even "known" systems. Most current adaptive filtering schemes are linear whereas this research uses a non-linear.

### Image Sequence Analysis

Nagel (1981) has written extensively on the applications of image sequence analysis. Among the applications he cites are video data compression, analysis of satellite imagery for meteorological and geological purposes, and temporal analysis for medical diagnosis and disease prevention. After discussing the various applications, Nagel (1981, pp. 163-164) comes to the conclusion that there are common threads in the need for improvements.

- quantitative models to describe complex phenomena
- lower-level image processing approaches
- general formulation of solution approaches vs. application-specific solutions

These needs are addressed by the current research. An anomaly is a change in the data stream relative to the model's expectations. This project is focused on reviewing data sequences and retaining unexpected portions, using a quantitative approach to modeling complex, dynamic, and staring continuous-dwell sensor data streams so that anomalies can be detected. By "unspecified" is meant that the research created a technology that is general and not very application-specific. The result deals with data streams at a very low level, source-by-source, with no correlation between the sources.

The following authors note that change detection is a research issue in the analysis of satellite sensor data. Ehlers and Shi (1996) identify dynamic change detection as one of the key problems in natural resource management. According to Aach and Kaup (1995), "The detection and accurate localization of intensity changes between subsequent frames of image sequences is a crucial issue for coding moving video, in particular for region-oriented techniques, as well as for a variety of tasks in image analysis". Bruzzone and

Serpico (1997) have a similar sentiment, "Detection of land-cover changes, by comparing the images acquired in the same area at different times, is one of the major applications of remotely-sensed images acquired by Earth-orbiting satellites". Abutaleb, Elfshawy, and Kesler report that, "Detection of targets (small changes in the gray levels of images) embedded in a background of interfering clutter plus noise is a very important task in applications such as remote sensing for monitoring growth patterns of urban areas, weather prediction from satellite images, diagnosis of disease from medical images, optical and infrared detection from radar images, and visual industry application".

### Intrusion Detection

Intrusion detection is an area where this research applies because an intrusion is a change in a scene or other signal that is not expected. The work in this area is represented by Smith, Peters, Curry, and Gupta (1998); Enomoto, Kawasumi, Ohata, Okazaki, and Sekii (1997); Kanemaru, Kaneta, Kanoh, and Nagai (1993); Lee and Stolfo (1998).

Finally, Kanemaru, Kaneta, Kanoh, and Nagai (1993) state that, "There is a need for fully automated [industrial television camera image] processing since, with present technology, detection relies solely on the judgment of personnel who monitor displays".

According to Chantry (1998), there is a growing interest in intruder detection systems that minimize false alarms. This is derived from people's desire to protect property in the face of sophisticated criminals who find it easy to defeat common alarm systems. Used as a passive intruder detection system, the technique should separate intruder events from dynamic backgrounds.<sup>2</sup> Postulated intruder events can be displayed to security personnel or submitted to further analysis. Chantry's (1998) article is about infrared detectors but the technique does not depend on the spectrum.

---

<sup>2</sup> The term "passive" is used here to mean that the sensor does not emit a signal or depend on a cooperative emitter, as an "active" sensor would. This is an important distinction since the presence of a passive sensor is much harder to detect. An undetected sensor is much harder to spoof. Speaking in an electrical sense, according to Taylor (1997, p 21), "passive" and "active" are used to denote sensors that are either powered exclusively by the measurand's signal or which deliver more energy than is drawn from the measurand.

## Implementation

Applications like NASA's EOS satellite constellation are becoming common. EOS transmits 50 gigabytes of data every hour, all day, every day of the year (Way and Smith, 1991). Ground stations have to analyze and store this volume of data for any value to be realized. Even relatively simple and slow-speed sensors such as the QuickCam produce 30 frames per second. Thus, a frame would have to be processed in less than  $1/30^{\text{th}}$  of a second (~33 milliseconds). Each frame can be sized at 640x480 30-bit pixels. Any method chosen to solve the anomaly detection problem has to be scalable to this size problem, at minimum. Solution options will likely have to be a combination of hardware and software. Choosing commercial off-shelf hardware would enhance the industrial transition of the result.

Finding a means of automatically modeling the infinite variety of scenes that a sensor could observe is critical to this research. Given traditional modeling methods, a solution may appear impossible. However, for instance, think of a digital image stream as a sequence of frames. Each frame is composed of a 2-dimensional grid of pixels. (The "grid" may be multi-layer but each layer is still a grid.) A pixel in a digital image stream can be thought of as nothing more than an independent single-dimension data stream. Sanner and Slotine (1991, May/November; 1991, December), while doing research in adaptive control theory, produced a time series analysis method that predicts the next sample of a one-dimensional numeric sequence. If a one-dimensional numeric sequence can be predicted, then it should be possible to predict what the next frame in an image stream ought to be since a stream of image frames is nothing more than a stream of pixels or one-dimensional data streams. Anything sufficiently beyond that expected in any pixel could be postulated as an anomaly. When viewed as an image, the detected anomalies would appear to be, for instance, intruders moving through the scene, evolving forest fires, or excessive pollution. Pushing the technology to these limits for improved anomaly detection in a practical sense is not straight forward. Chapter 1, Barriers and Issues, discusses the related problems.



## **Practical Applications of the Solution**

The author's modus operandi has always been based on the premise, "No theory without practice. No practice without theory." It is insufficient to develop theory without ensuring its practical implementation. Just as important is that all implementations should have a solid grounding in theory. Basic research is something that should always be funded. Such research provides the concepts, mathematics, and tools we will need before we know we need them. However, it has been this author's experience that such endeavors need to ride on the platform of practical application. It is practical application that attracts the funding for basic research. (Additional discussion can be found in Raeth, 1993.)

Since the ultimate benefit of technology lies in its practical application, this section identifies just a few published success stories. These serve to illustrate the ways adaptive systems are providing value in the near term as an encouragement for additional work that will advance the technology further. There are two basic categories of application: business/management and technical/engineering. Emphasis is on the latter since that is the author's primary development and insertion focus. For additional insight into the former see the following:

### Business/Management

- determining opportunities for cross sales
  - (Anand, Patrick, Hughes, and Bell (1998))
  - (Holstein, 1998)
- process for business data mining
  - (Brachman, Khabaza, Kloesgen, Piatetsky-Shapiro, and Simoudis (1996))
- a research perspective
  - (Kersten, Siebes, Holsheimer, and Kwakkel (1993))
- business benefits
  - (Sharman, 1997)

### Technical Applications In Engineering

Mango and Auriol (1998) introduce two applications that apply case-based reasoning and inductive learning to generate maintenance applications based on previous experience. The first application is by Cfm-international, a partnership of General Electric and Snecma. This application provides maintenance advice on the Cfm 56 jet engine family for Boeing and Airbus aircraft. 23,000 maintenance cases were extracted from a historical database. After validation and cleanup of these cases by master mechanics, the cases were used to automatically generate a fault tree. This tree can be updated as new cases are validated. The fault tree is the heart of a process that queries the user as to symptoms and then provides suggestions on possible causes and tests to perform. The process is linked to the Illustrated Parts Catalog for visual parts cues, automatic updates related to system evolution, and statistics gathering on reliability and maintainability from the world-wide community of users. This system helps to mitigate the 50% of total engine downtime that is due to waiting for faults to be diagnosed.

Second is an application in troubleshooting robot axis positioning. Sepro Robotique develops robots that automate the process of grabbing plastic parts from an injection molding press and manipulating them in various ways. Each robot is customized for a particular purpose. Axis position defects represent about 30% of the problems that are the most difficult to solve. Automated diagnostic advice starts with a fault tree built from 150 historical cases. About 10 cases a month are added to the fault tree. As a result, the system is able to resolve 75% of the cases presented to it.

Other applications are presented in the following citations:

- machine condition monitoring
  - (Peck and Burrows, 1990)
- determining system behavior envelopes
  - (DeCoste, 1997)

- capturing transients and trends in scientific data
  - (Hinke, Rushing, Ranganath, and Graves (1997))
- modeling industrial blasting
  - (Chapman, Bollinger, and Sibol (1992))
- correlation of waveforms
  - (Israelsson, 1990)
- training sample labeling
  - (Hsieh and Landgrebe, 1996)
- learning from manufacturing systems
  - (Koonce, Fang, and Tsai (1997))
- equation discovery
  - (Bharat, Lu, and Stepp (1990))
- scientific data mining
  - (Chang, Wang, and Chang (1996))
  - (Kapetanios and Norrie, 1997)
  - (Fayyad, Piatetsky-Shapiero, and Smyth (1996))
  - (Fayyad, Haussler, and Stolorz (1996))
- throughput screening
  - (Holland, 1995)

## **Implementation Technology**

### Neural Networks

A technique for finding if-then rules in a set of observations are discussed by Zhong, Dong, and Ohsuga (1998): Generalization Distribution Table using neural networks and rough sets. These tables provide a probabilistic basis for determining the strength of a rule. Neural networks and rough sets facilitate the use of data that is uncertain and incomplete. This is an important ability since data in the real world is not perfect and can not be. This present research does not develop rules but the integrated discussion of data mining, neural networks, and real-world data provides a lot of useful points.

Agehed, Eide, Lindblad, and Lindsey (1998) give a valuable paper that talks about creating a categorization of images in a large database. These categories could lead to a retrieval index based on image classification. They also discuss the implementation of neural networks via look-up tables. Their intent is different from my own but the thought may yield ways of increasing the throughput of the model. They are particularly keen on hardware implementations via reprogrammable devices. They also realize that images mean data volumes large enough that even post-collection processing needs methods that have improved speed over those that would be satisfactory for single-dimensioned data. The weakness in their argument is that they only postulate speed improvements from hardware implementations without providing actual test results. Such results are needed to form a judgment on cost vs. time.

Goodwin (1994) gives a brief overview of the advantages of neural computation over traditional methods and applies those advantages to sensor data processing. Of particular interest is the discussion of chips that are specially designed to accommodate neural architectures. The chips he discusses are no longer on the market but one does get a good idea of what features implementation hardware should meet. A primary problem with this present research is the size of the model required for minimally-useful real-time image stream processing. The model can be likened to a neural architecture so this article gives some encouragement that a model can be implemented that effectively detects unspecified events in natural image streams, and that performs this task in real-time for meaningful frame rates and resolutions.

### Basis Function Networks

There is precious little in the literature at this point that applies neural networks to data mining. (In many ways, the present research target can be cast as a problem in data mining.) The paper by Lu, Setiono, and Liu (1996) concentrates on classification whereas this present research is on function approximation. However, he has a good introduction to neural nets in data mining. Seeing his approach is worthwhile. Another helpful part of

this paper is the comparison of the neural network rules to those generated via the C4.5 decision tree approach. This fed my own thinking on metrics. Finding metrics that can be implemented with natural data for the large data sets that make up image streams is not simple.

Tan, Hao, and Vandewalle (1993) apply radial basis functions to the identification of non-linear systems. Of particular interest is their discussion on updating the functions' parameters. They lend insight to possible improvements to the method currently in use by this research.

Chen, Billings, Cowan, and Grant (1990) explore the representation capability of networks of radial basis functions. They explore the conditions necessary for functions of one variable to be qualified as activation functions. They also present experimental results of attempts to approximate nonlinear systems in the frequency and time domains. Their results indicate there are alternatives to using the gaussian in this project's basis function model.

Chen and Chen (1993) explain and demonstrate a fully automated means of choosing the centers of the basis functions and adapting their parameters. Their application is in general time-series analysis, a particular area of this present research. Several approaches are discussed. An important proof here is that a necessary and sufficient condition for a function of one variable to be used as a basis for forming larger models is that the function not be an even polynomial. This leads ultimately to a large number of choices besides the radial gaussian currently being considered by the present research.

Peck and Burrows (1990) wrote a valuable paper that compares expert systems and neural networks as applied to real-time equipment monitoring. They also provide a top-level architecture for how the whole process from collection to fault detection and identification might work. Their approach is very much like what is needed in image sequence analysis and event detection for real-time response and post-collection indexing. One fault in the paper is the processing architecture's comparison of expert

systems and neural networks. The authors make the neural network approach look simpler than it is, as if there is comparatively much less to do. Having had experience in both, this author disagrees with that notion. A strength of their paper is the very practical approach taken and the inclusion of traditional analysis methods such as Fourier Transforms to facilitate data preprocessing.

Brown (writing in the book edited by Light, 1992, pp. 203-206) proved that if  $D$  is a compact subset of the  $k$ -dimensional region  $R^k$  then every continuous real valued function on  $D$  can be approximated uniformly by linear combinations of radial basis functions with centers in  $D$ . This is a fundamental proof that lends credibility to attempts to model complex systems using radial basis functions. Similar proofs have also been generated by Funahashi (1989); Girosi and Poggio (1989); and Hornik, Stinchcombe, and White (1989).

Supportive work in the area of using basis functions for data fitting was accomplished by Watson (writing in the book edited by Mason and Cox, 1987, pp. 337-356). He concentrates on the two-dimensional problem of observation  $y$  taken at position  $x$ . His methods may be extendable to the problem type where  $x$  is time.

Raeth, Gustafson, and Little (1994) employed gaussian radial basis functions for classification. This work provides basic technology for the present application. The research being pursued here employs gaussian radial basis functions in a much different way but the mathematical and software approach is derived from this earlier work.

Raeth, Gustafson, Little, and Puterbaugh (1992) applied gaussian radial basis functions to the problem of  $k$ -dimensional interpolation. This work also provides basic technology for the present application in that the authors' later work in classification was derived from this work on interpolation.

Powell (writing in the book edited by Mason and Cox, 1987, pp. 143-167) made contributions to multivariable interpolation. His work is very theoretical but does lend some insights that broaden one's understanding of how to use radial basis functions.

### Time-Series Analysis

Sanner and Slotine (collected works) provide the basic theory for the present research. Their application is in adaptive controls for automated machinery. However, their thrust into the prediction of numeric sequences has wide application.

Weiss and Indurkha (1998) specifically address a topic within the present research, time-series prediction. Data preparation is a strong part of the material as is a comparison of traditional and adaptive approaches. There are some good procedures here for data preparation, reduction, modeling, and analysis. The authors take care to lead the reader through some actual examples. They also include a software tool that can be used to implement these examples. There is sufficient information and generality that the tool is not essential to getting the best value from the book.

DeCoste's (1997) paper is directly related to the present research in that it seeks to model a sensor's data stream so that future anomalies can be detected. Their technique works with multi-dimensional data and makes no claim to real-time potential. However, there is clear post-collection capability. His basic technique uses regression, a process of collecting a batch of data and then working backwards to create a predictive model. Although the present research also creates a predictive model built from past results, it is not a regression technique. It does not collect a batch of data and then perform some calculation. Rather, it postulates a model and then adjusts that model as each new sample arrives. In this sense the model incorporates ALL past samples rather than a fixed number. From this point of view, the technique is more akin to the mathematical technique 'regula falsi', "a method of approximating an unknown quantity by first making an estimate and then using known properties of the quantity to improve the

estimate" (Morris, 1992). The known property is the next sample. The estimate is the model's guess at what the next sample will be.

Chen and Chen (1993) explore the representation capability of networks of radial basis functions. They explore the conditions necessary for functions of one variable to be qualified as activation functions. They also present experimental results of attempts to approximate nonlinear systems in the frequency and time domains.

In their two papers, Chen, Billings, Cowan, and Grant (both in 1990) explain and demonstrate a fully automated means of choosing the centers of the basis functions and adapting their parameters. Their application is in general time-series analysis.

### Tools

Anandan and Shafer (1999) have developed a software tool kit for vision experiments that is freely available to researchers. This is not a finished system but a collection of Visual C++ software modules that can be used in systems being built by others.

Akin and Sanner (1990) discuss computer architectures as part of their efforts to get basis function networks to perform in real-time vehicle control applications. They show that their approach would be effective if appropriate hardware architectures were employed. This paper was useful as this project moved into processing natural images.

Chantry (1998) is the managing director of a company that produces intrusion detection devices. His article explores many of the misconceptions that people have when they install these devices. These misconceptions lead to mistakes in installation and unmet expectations. In all, it is an extremely useful and practical article that aids the effort to develop new capability for this market.

Eberart and Dobbins (1990), and Masters (1993), discuss measures of error for neural network calculations. This research employs networks of gaussian radial basis functions



that can be cast as neural networks. Thus, there was a lot to learn from these authors on how to measure detection effectiveness.

Logitech Engineering (1999) is a practical applications company. They sell various grades of cameras known as QuickCams. These are for home use, at the low end, and teleconferencing use, at the high end. Their datasheets are the source of technical specifications for cameras intended for use in this project.

Smith, Peters, Curry, and Gupta (1998) explore discrete event simulation as a means of evaluating intrusion detection systems. Their basic thrust involves the modeling of floor plans, the placement of detection devices, the movement of intruders, and the movement of security personnel. This work helps to lay a framework for judging the value of new data analysis techniques as an adjunct to established security approaches.

Suvan (1999) is a student at Northeastern University working in machine vision. He has produced an excellent tool for real-time image capture and processing that he has released for use by researchers. Using a pipeline approach, the experimenter can employ the provided techniques or install unique ones. Full documentation and executable software can be downloaded from the URL provided in the reference list.

## Metrics

The main value of Masters' (1993) book is its discussion on measures of error (actual vs. expected output from the system being observed). This present research uses an error measure derived from those presented in this text. There is an additional discussion on confidence measures, time-series prediction, and function estimation. Another chapter on data preparation also provides useful information. If one is just starting in neural networks, this is a very reliable book. It covers all the major bases and gives complete examples, including source code. Like most writings on using neural networks for time-series prediction, Masters (1993) uses  $x$  previous samples as input. In my own experience, this increases the size of the network far more than does the single-input, single-output approach. Additionally, Masters (1993) includes a discussion on trend

elimination and "seasonal" variations. This is not necessary in the approach employed by the present research since the network evolves as the data arrives. In this approach, a complex function is constructed that takes into account all previous data, not just the last  $x$  samples. The size of the function is based on the expected frequency content of the data.

If you need to apply statistical measures but are not a statistician then look to the writings of Bruning and Kintz (1968). This present research needed a measure of statistical significance. This book presents such a measure complete with a computational process and examples. While most texts concentrate on statistical theory, this book concentrates on application. The result is that the reader understands how to apply the given statistical test and how to interpret the results. Covered are tests like mean, standard deviation, standard error of the mean, t-tests, analysis of variance, correlation, nonparametric tests, tests of significance, and indices of relationship.

## **Reference Material**

### Specialized Dictionaries

Morris' (1992) dictionary consists of 2500 pages and 133,000 entries. It is a general technical dictionary.

Cubberly's (1988) dictionary contains 9000 terms and 350 pages. It is specific to instrumentation and data acquisition. It provides terminology bearing on the engineering applications of this present research.

### Data Extraction and Preparation

While this present research makes no assumptions about whether the data is from an image, audio, or other stream, Davies' (1997) book does give some useful insights on noise removal. It presents accumulation, threshold, and gaussian filtering methods. For practical purposes, it eventually will be necessary to make use of domain information to

achieve a commercial result from this research. However, this author desired to push the general ideas behind prediction, adaptive thresholds, and event detection to their limits to see how far they can go before domain assumptions are necessary.

Fayyad, Haussler, and Stolorz (1996) have written an article that discusses reduction of scientific data and the process of going from extreme amounts of raw data to conclusive results. The authors see the automation inherent in data mining as a way of overcoming the growing gap between data volume and human analysis capacity. An important part of their article is the discussion on various types of data and the means of dealing with them. Fayyad also presents two case studies.

In his article, Fayyad (1997) expands his thinking on the application of data mining to extracting value from scientific data. He talks about the increased value as one moves from databases to OLAP to data mining. Best value from the article comes from material on the different ways to extract value from scientific data. He talks about predictive modeling, clustering, data summarization, dependency modeling, as well as change and deviation detection. As such, there is strong input to the present research.

There is A LOT of data involved in the present research. Just one short collection with a QuickCam set to low resolution and slow frame rate can result in 3gb of data. Thus, it is necessary to think carefully about data storage and processing. The paper by Hinke, Rushing, Ranganath, and Graves (1997) goes a long way in discussing these issues. Additionally, they deal with satellite data, a source likely to be encountered eventually by this present research. Generally, their paper takes large volumes of data stored on tertiary devices and summarizes it so it can be placed in a database on primary storage. The justification given is that the new version of the data is far smaller in volume but much higher in value. Generating trend data and outliers is the basic approach used. Calculations are used to show that the accuracy of the resulting summary does not vary much from the original data in the sense of statistical significance.

### Supporting Information

Russ (1999) produces an image processing handbook that has been through numerous reprints. It is an accepted standard text that gives well written explanations of technology topics. Practical application is its guide but attention to theory is given where needed. His main focus is on the processing of single images for display, printing, and multimedia applications. As such, it is a good overview of the issues to consider when processing a scene's individual frames.

Schalkoff (1989) has written quite an informative basic text on image processing. One of his best contributions is his discussion on the difficulties of practical applications. By now, his book is somewhat dated but his clear explanations of basic technology issues are timeless.

Gong, Ledrew, and Miller (1992) introduce and evaluate two methods for reducing the registration noise in two images of the same area taken at different times. Their methods help to ensure that the two images actually overlap in a spatial (geographic) sense. Their discussion is not directed at continuous image streams but rather to multiple passes over the same area during an extended period of time. However, the idea of imperfect registration also impacts this present project due to the vibration expected in the camera's platform (jitter).

Aach and Kaup's (1995) thrust at change detection involves Bayesian statistics. Their work aids this project mainly by discussing the drawbacks to existing change-detection systems. They also discuss the need for adaptive systems in this area of work.

Bruzzone and Serpico's (1997) work is highly specialized to multispectral sensors. Their input may prove useful if this technology is applied to satellite imagery. Although this project will not delve into that area, their general discussion does provide some basic education that is useful.

### Change Detection

The title of the article by Kanemaru, Kaneta, Kanoh, and Nagai (1993) almost causes it to be skipped. Instead of referring to "power line towers", they should have referred to "highly complex scenes". Their piece is an excellent discussion on the difficulties encountered in performing intrusion detection in natural scenes. They also talk about the use of industrial cameras for such applications.

Shi and Ehlers (1996) give a good introduction to the sources of error in detecting changes in sequential images. They identify dynamic change detection as one of the key problems in natural resource management. They also talk about another approach to change detection, analysis of multi-date composite images. Their basic thrust is to overlay multiple images and then to analyze the changes in color hue or grey tone.

Mas (1999) discusses six methods for detecting changes in images taken of the same area over an extended period of time. Although his focus is on the best use of specific satellite-gathered spectra, his discussion on change detection does prove useful. Overall, he finds selective principal components analysis to be best in his application for detecting image changes. This approach is also used by image analysts interviewed by the author.

Elfishawy, Kesler, and Abutaleb (1991) offer two approaches to the detection of small changes in a pair of images. They apply variations on least squares to accomplish their goals. Their discussion of problems and issues bears directly on the author's research. The alternative approaches may prove useful when the reality of natural scenes strikes this project.

Kerestecioglu's (1993) approach to change detection departs quite a bit from that intended in this project. However, he has written an excellent article that introduces the subject and explains its importance in system and process control.

## Chapter Three

### Methodology

This chapter details the experiments used to demonstrate the effectiveness of the approach chosen to detect unspecified anomalies in staring continuous-dwell sensor data streams. While certain data streams were specified, the detection model was not tuned to the stream based on the source. Some loose assumptions were made:

- min/max of the expected actual values
- general robustness and spectral content as illustrated via Fourier Transform analysis (although no anti-aliasing filter was applied to the data)

Presented first is an overview of the author's approach to projects in adaptive automation. This foundation guided the specifics of the individual experiments.

### A Philosophy of Adaptive Automation Development

In beginning this discussion on Methodology, it seems best to present a general philosophy that supports adaptive automation projects. An understanding of this philosophy explains why certain choices and steps were used to conduct the research.

#### Working Definition

Adaptive Automation is a systematic process of finding meaningful new correlations, models, patterns, and trends by sifting through huge volumes of data. Far beyond only confirming preconceived hypotheses or organizing the results of queries, it is a process of extracting hidden, non-obvious, information that human experts or conventional approaches could miss because it lies outside their sphere of consideration. Adaptive Automation allows business managers, system analysts, scientists, engineers, and automated analysis/response mechanisms to make proactive decisions informed by objective observations.

## Process Review

When developing credible adaptive automation for post-collection or real-time environments, having a clear and objective process is essential. Quite contrary to the impression given by too many speakers and writers, you can not just take your data and throw it into some black box without regard to analysis method, theory, data quantity and quality, experimental design, or domain knowledge. A mistake beginners often make is to think they can just “plug and chug” with some analysis tool. One trap such people fall into right off is the one described Fayyad, Piatetsky-Shapiro, and Smyth (1996, p 27), “If a system tests  $N$  models at the 0.001 statistical significance level then, on average, with purely random data,  $N / 1000$  of these models will be accepted as significant.” Other traps the author has seen are the following:

- certain problems’ high dimensionality ( $D$ ) and the analysis model’s size growing as does  $C^D$  (where  $C$  is some constant relating to the problem’s size and complexity), along with a corresponding growth in memory and CPU utilization
- failing to account for insufficient, missing, noisy, incorrect, or uncertain data
- problems where each dimension has a data range widely different from the other dimensions
- failing to pre-process data that is closely packed in the domain space
- failure to carefully conduct method selection, construction, testing, and validation with the result that, for instance, clustering methods yield false classifications for critical known cases in the training data

Given the challenges and traps of adaptive systems, one should establish a formal process for conducting a project. Several authors discuss adaptive systems development processes. Fayyad, Piatetsky-Shapiro, and Smyth (1996, p 10-11) give the most concise and complete list of process steps. Bigus (1996, pp. 43-60) offers an excellent discussion of data preparation. Look to Groth (1997, pp. 12-22) and Kennedy, Lee, Reed, and Lippmann (1998) for larger perceptions on the process steps. The following discussion is a summary of these sources with expansion due to the author’s experience.

Any worthwhile adaptive systems development process should contain the following steps. Although these steps speak to the “systematic” part of the working definition of adaptive automation, the reader will note that there are algorithmic and heuristic components. These are needed to accomplish the analytic parts of the process.

1. Know the end user's goals and have access to domain knowledge
2. Choose the development task
3. Select an appropriate data set
4. Choose the algorithms and heuristics
5. Clean the data
6. Perform appropriate data reduction and transformation
7. Apply the selected algorithms and heuristics to accomplish the task at hand
8. Interpret and validate the results
9. Iterate through the above steps as needed
10. Consolidate, review, and document the results
11. Install the results in the target system

Lets talk about each one of these steps.

#### *Know the End User's Goals and Have Access to Domain Knowledge*

There is no substitute for knowing the domain of application. A technical specialist in adaptive systems will do well to develop a solid relationship with a domain expert. Such a combination of talent can be invaluable to an organization. A background in the domain will enable the successful accomplishment of the other steps in the adaptive systems development process. Once domain expertise has been established, find out who the real end user is. Sometimes the end user is the domain expert. Sometimes it is someone who performs day-to-day tasks. Consider the question, “Who is the domain expert's end user and what is that person's goal?” Honest consideration of this question and attention to the answer can result in continued funding for the project. The author's experience is that the goal of all enterprise in our society is profit. Sometimes “profit” is measured in terms of additional money after expenses. Sometimes it is measured in other terms such as more lives saved in dangerous situations, additional crops harvested, more efficient energy



distribution or usage, lowered maintenance cost, improved material purity, lower ambient noise, and so forth. The result of any adaptive systems project needs to be couched in terms of such improvements to the bottom line. If you do not have ready access to domain expertise or can not determine what the improvement to the bottom line is, be careful about expending any serious funding. Forward thinking companies encourage exploration but, once serious funding is expended, serious results are expected. So, to maintain your credibility, stay away from over promising and under delivering.

### *Choose the Development Task*

Having the required expertise is the first step. Having a clear idea of what it is you are trying to accomplish, why its accomplishment is important, and how you are going to accomplish it is next. What is the problem you are trying to solve? What is the improvement you are trying to make? By when does the solution need to be available? Who and what is needed for the task to be successful? Fayyad, Piatetsky-Shapiro, and Smyth (1996, p 24) list criteria by which a good adaptive systems project may be identified. These can be expanded as:

- potential for significant positive improvement according to the organization's success criteria
- no good alternatives exist within more traditional analysis methods
- solid organizational support exists
- there are no privacy or other legal issues outstanding in the use of the data or in the production of the intended result
- sufficient data can be obtained
- the available data is relevant to the intended project
- the available data can be cleaned so that it has an acceptable level of accuracy and completeness
- objective criteria exist for judging the project's results

- a domain expert exists, is committed to the project's success, and can dedicate a sufficient amount of time to the project

These may sound like simple things but there is no substitute for a clear definition of and support for the task at hand. Write the goal down. Define in definite measurable terms how you will know when the goal is reached. Have a plan with milestones that takes matters from beginning to end in time to make a difference. Something that has worked well for the author is to under sell and over deliver. The idea is to understate what the likely result will be and when that result can be accomplished. Further, be aware that sometimes a lesser solution is better than no solution at all. Always strive to deliver some measurable improvement in the near term while building toward a larger long-term solution. Recognize too that it is not necessarily a bad thing to aim at one target and hit another. Just be sure to show a positive impact on the bottom line relative to the funds expended.

### *Select an Appropriate Data Set*

Finding the right data is the next step in the adaptive systems development process. It takes a domain expert to make this determination. This person will have a deep understanding the data's source and meaning. Here is where the input and output variables (data elements or fields) are selected. This is also the chance to determine representative subsets of the input data records. Input records must be selected for the modeling/training step, testing step, and validation step. Additionally, the practitioner has to decide if there is sufficient data quantity to carry out the development task. Solutions to the problem of insufficient modeling data include recycling the data until the model is completely built, using simulation to generate additional data, and putting in electronic form data that was collected manually. None of these are pristine solutions but, used judiciously, they can sometimes yield implementable results or at least allow you to demonstrate a proof of concept.

### *Choose the Algorithms and Heuristics*

In many ways, the choice of choice of algorithms and heuristics needs to be done in synergy with the choice of data. For instance, some methods are more tolerant of errant data than others. Having more or less data volume will make a lesser or larger difference, depending on the method chosen. Supporting technologies are discussed in the next section.

### *Clean the Data*

A common assumption that must be made about the data to be used is that it will contain a certain percentage of dirty records. For instance, according to Fayyad, Piatetsky-Shapiro, and Smyth (1996, p 27), U.S. census data has error rates of up to 20%. Dirty records take the form of inaccurate values, missing values, and other inconstancies. Adaptive processes will only yield results that are as reliable as the data feeding the process. Thus, the data must be as clean and correct as possible. There are several methods that can be used to accomplish this. Rule-based techniques evaluate each field according to an expected range or list of values. According to Simoudis, Livezey, and Kerber (1995), rules are also useful for examining the relationships of one field to another. Visualization is another useful method of data cleansing. This approach can be used to identify outliers. Statistical processing can be used to set missing or incorrect values to valid, or at least neutral, values. If noise exists in the data, there are some good filtering techniques that may prove useful (see Embree (1995) for instance). Noise is mainly due to measurement and collection errors that are not caused by human error. They are due to the limitations of the system being measured, the path across which the data must travel, and the data collection equipment.

### *Perform Appropriate Data Reduction and Transformation*

When the data is as complete and accurate as possible, it is time to enhance the data. Interestingly enough, there is not necessarily a direct relationship between data volume and information content. There could be elements that are simply duplicates or computed from other elements. A decision is needed on which elements to eliminate. On the other hand, new elements may need to be created by preprocessing single elements or by combining several elements. Here again, a decision has to be made on whether to eliminate the root elements. By and large, success and the time it takes to apply the selected algorithms and heuristics depends a great deal on data reduction and transformation. Most guidance on this subject depends on the domain application and the algorithms and heuristics chosen. Hence, further discussion on the methods of data reduction and transformation is beyond our scope. Look to Bigus (1996, pp. 49-55), Bevington and Robinson (1992), and Weiss and Indurkha (1998, pp. 51-118) for some excellent insights.

### *Apply the Selected Algorithms and Heuristics to Accomplish the Task*

This is where the search for patterns of interest in particular representational forms takes place. Getting through this step successfully depends on how well the preceding steps have been performed. According to Kennedy, Lee, Reed, and Lippmann (1998, p 11.4), these steps are useful:

- Divide the available data into training, test, and evaluation sets
- Set values for user-selectable parameters in the algorithms and heuristics
- Use the training data to adjust the method's variables
- Use the test set to determine if the variable settings continue to hold
- Repeat the above steps using different parameters
- Select the best set of parameter values
- use the training AND test data to adjust the method's variables
- validate the end result by applying the evaluation data set

### *Interpret and Validate the Results*

Interpretation and validation imply much more than calculating an error factor. The results have to be cast in terms of the application domain and demonstrate a chance to create a meaningful positive delta relative to the bottom line. Here is a place where the domain expert has another serious role. An important question to be answered is, "Do the results make sense?" Some results will be absolutely correct but not intuitive. Other results will have clear statistical significance but be total nonsense. The difference can be subtle. Careful effort must be applied so that it is possible to see the difference. On the one hand, this is an opportunity to separate the new opportunities that are revealed from the worthless results. On the other hand, this is a chance to "think out of the box". Here too is a place where debugging of the method's use should be performed. Each method has its own nuances and so offering debugging assistance is beyond the scope of this dissertation. However, some useful ideas are given by Kennedy, Lee, Reed, and Lippmann (1998, pp. 12.14-12.26).

### *Iterate Through the Above Steps as Needed*

At the risk of sounding repetitive, let it be emphasized once again the importance of the domain expert. Anand, Bell, and Hughes (1995) offer additional discussion on this topic beyond that given above. They say that the domain expert can offer information of two types:

- information about the data that is already available either through some other process or as part of the domain's common knowledge
- attributes of interest within the data, support and uncertainty bounds, opportunities for bottom-line improvement, and identification of unsolved problems

Information about the data generally falls within three classes:

- Hierarchical Generalization Trees - generalizations of attribute values that may be of interest to the end user
- Attribute Relationship Rules - integrity constraints defined within the design of the database or information already known with high certainty
- Environment-Based Constraints - constraints put on the adaptive process by the domain itself

If you have gone through the adaptive systems development process once without the help of a domain expert, you will likely have realized the need for one by now. Do recruit one before trying the process again.

Once you have generated a baseline model, it is time to try to improve the results. Examine your work in each of the process steps. Try to improve each one individually and then put them all together. Beware of simply collecting more data. Many practitioners expect that substantial amounts of new data should lead to better performance. This is sometimes the case but do consider the following questions from Weiss and Indurkha (1998, pp. 112-113):

- is the error getting smaller in the statistically-significant sense
- is model complexity increasing much faster than the error is decreasing
- is the model's increase in complexity worth the smaller error

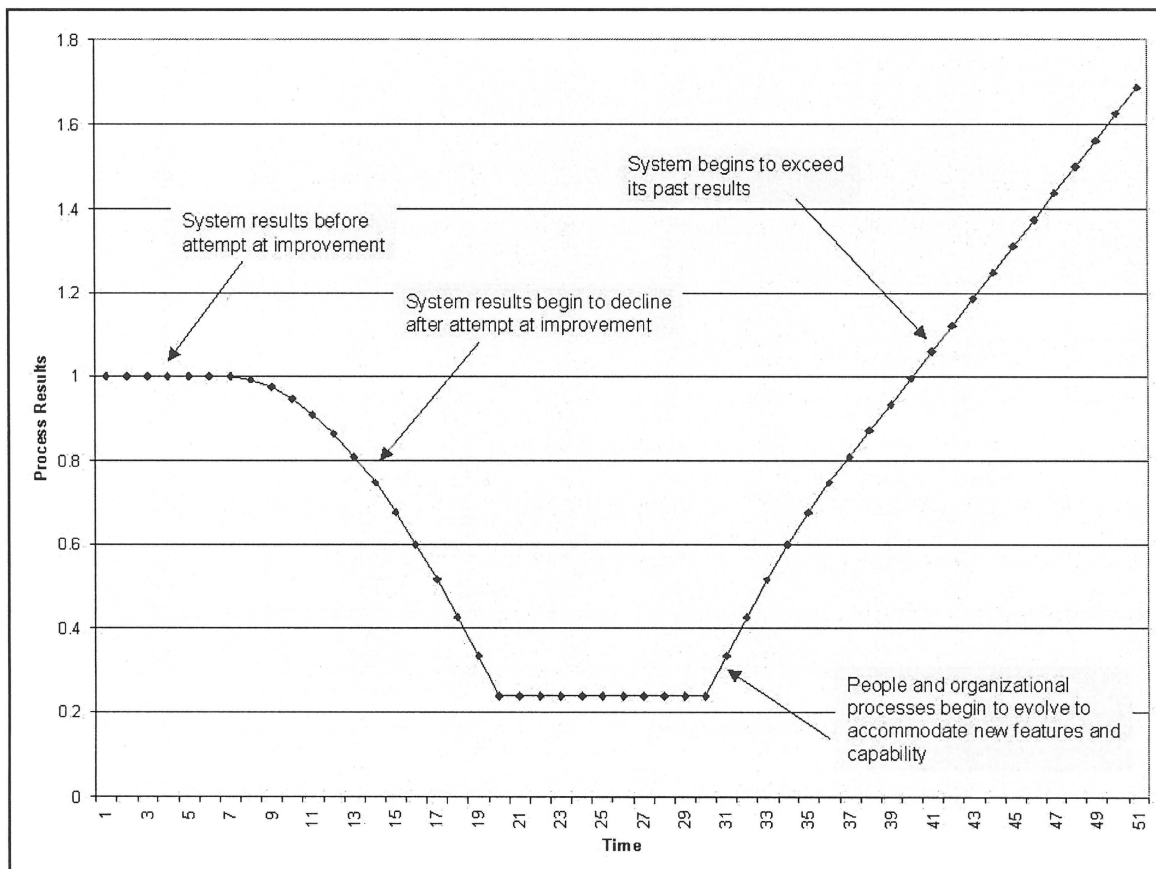
Developing effective and profitable adaptive systems is an art. This is told by the fact that most adaptive systems methods are heuristics rather than algorithms. They give approximate results relative to the actual reality. The actual reality itself is often not knowable. Adaptive automation is also a process. While the performance of the process steps is important, these steps are not discrete with no relationship to each other. Each must be seen as part of a larger whole. Additionally, the results of conducting one instance of the process has to be combined with the results of other instances in order to produce significant achievements.

### *Consolidate, Review, and Document the Results*

Consolidation gives you a chance to find and resolve potential conflicts with previously believed or extracted information. Careful records should be kept on each experiment. These records should enable another practitioner to repeat the exact process and duplicate the results. Records should be kept in a standard format to facilitate comparisons. Opportunities to publish in the peer-reviewed literature should be considered.

### *Install the Results in the Target Environment*

A classic performance curve that results from trying anything new in an established environment is shown in Figure 4.



**Figure 4. Expected Performance Drops at First With Attempted Improvements**

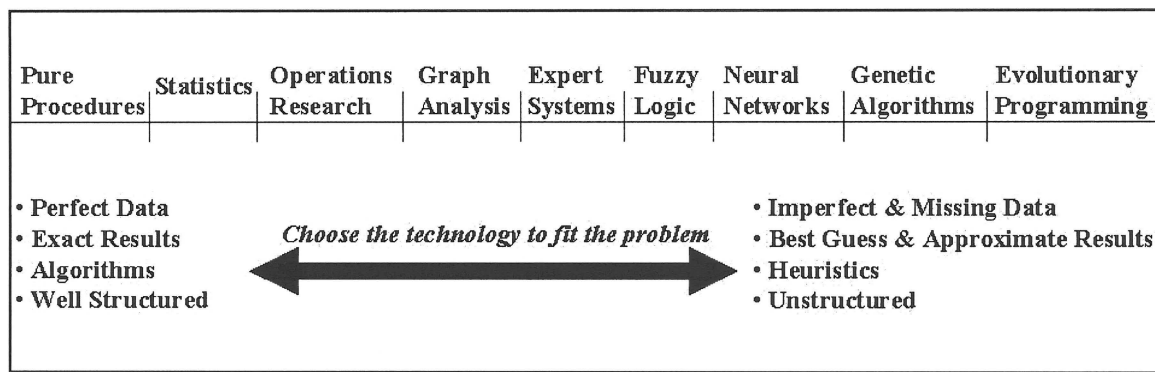
At issue with this curve is that an initial decrease in results is expected anytime something new is tried. The hope is that the system results will recover as people and processes evolve to accommodate new features, and that the new system will ultimately exceed past performance. Practitioners need to encourage the organization so that it understands the risks and to conduct the installation in such a way that overall profitability is not too negatively impacted. Here again the domain expert can assist in telling a potential improvement from a potential failure. A business mentor can offer advice on the impact of risk on the organization and for how long and how much a negative impact on the bottom line can be tolerated. The use of prototypes is highly recommended as is running the new system in parallel with the old for a period of time. Two worthy books on integrating new technology into established applications are written by Gillies (1991) and by Parmee (1998).

### Supporting Technologies

The previous section discussed a process for carrying out an adaptive systems development project. This section reviews technologies that could be used for carrying out the analysis part of the process.

An article by Pass (1997) touches on the idea that in any adaptive systems project there are requirements (what needs to be done) and technologies (how to get things done). The link between requirements and the technologies needed to meet those requirements is an important one. Adaptive automation technologies can be seen as lying along a continuum (Figure 5). On the one end are pure procedures. On the other end are the various methods of evolutionary computation. As one goes from one end to the other, algorithms (exact answers) evolve into heuristics (best guesses). Algorithms are for well understood, highly structured problems. Heuristics are better for problems that are not well structured or understood. They are also more tolerant of imperfect data.





**Figure 5. Adaptive Automation Technology Continuum**

Here are short definitions for the basic technologies in the continuum. These are expanded from Morris (1992).

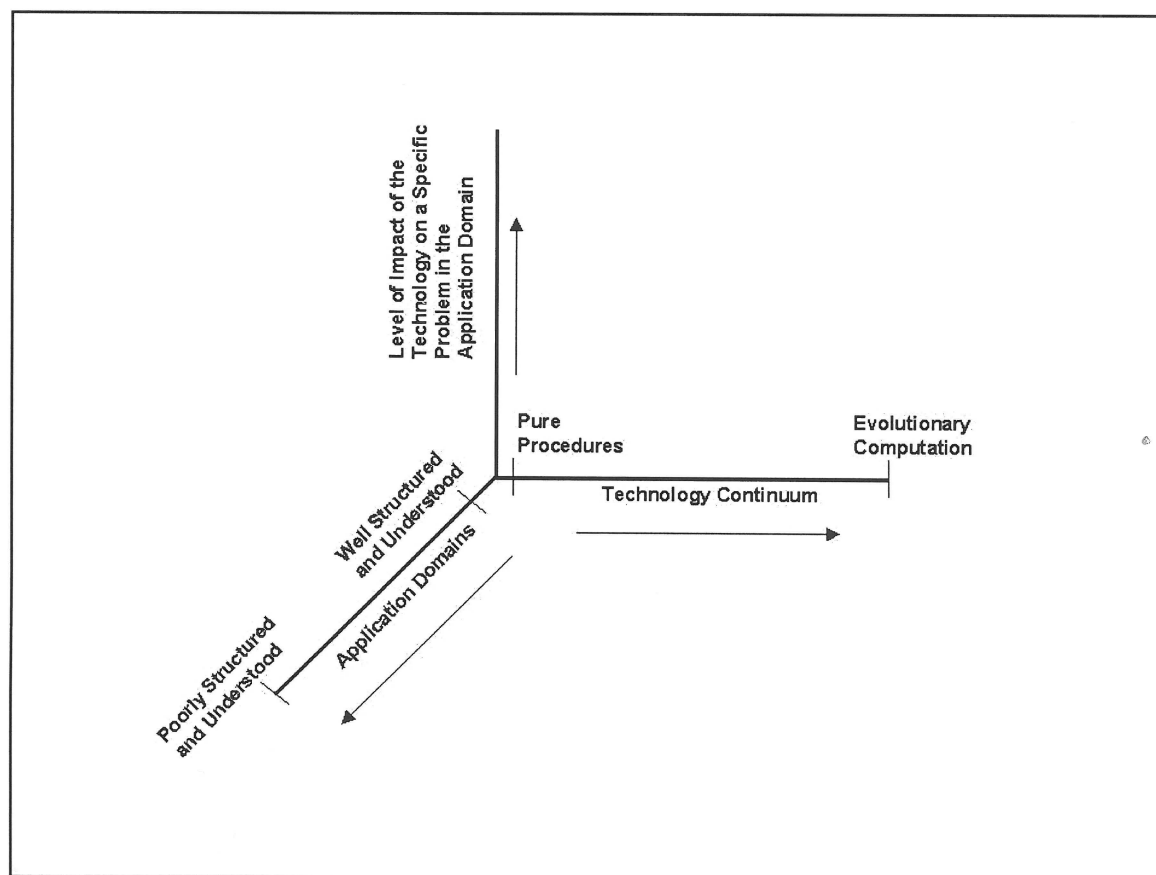
- Algorithms - well-defined unambiguous procedures for performing operations that lead to exact solutions in an efficient manner. Although adaptive automation is more closely associated with heuristics, algorithms can be used for preprocessing and to acquire insights into the data.
- Statistics - mathematical methods for estimating, confirming, or denying population parameters from a sample with varying degrees of accuracy. Adaptive automation does go beyond statistics, as stated earlier. But, statistics can be a worthwhile start for determining basic characteristics of the data.
- Operations Research - mathematical analysis of optimization problems, especially of dynamic systems subject to a given set of constraints. The purpose of such analyses is to evaluate the effectiveness of a given system in achieving identified goals.
- Graph Theory - the study of graphs, trees, and networks. A graph is a diagram or curve representing varying relationships between sets of data. "Data structures" is another topical area in this same genre. It is an organization of

data in some characteristic way, such as graphs, arrays, trees, lists, queues, networks, or stacks.

- Expert Systems - applications that solve problems by means of an information base containing rules and data from which inferences are drawn on the basis of human experience and previously encountered situations. Adaptive automation is often put to the task of deriving rules from data. However, there is no reason that previously established rules can not be used to guide a new processing session.
- Fuzzy Logic - a method of machine reasoning that can process uncertain or incomplete information. This system of logic is based on the fuzzy set, a generalization of traditional two-state set concepts. Membership in a given set can take on the entire analog range of 0 .. 1 inclusive, rather than just the binary discrete values of 0 (not at all a member of the set) and 1 (entirely a member of the set and no other). Fuzzy set theory applies these concepts to expert systems and estimates the degree of a conclusion's certainty.
- Neural Networks - a computational network composed of mathematically defined elements that are loosely based on biological neurons. These networks are often composed of one layer that receives and organizes the n-dimensional data input, a subsequent data processing layer, and an output layer in which individual neurons identify particular patterns or deliver an analog result as an estimate of the system represented by the data. There are numerous neural network types.
- Evolutionary Computing - a class of heuristics that attempt to solve a problem by means loosely based on natural evolution. A set of candidate solutions is evaluated in terms of solution quality. Then, the better candidates are allowed to reproduce (create modified duplicates that continue evolving). If one

candidate comes to dominate the population, it is selected as the best solution. At times, different candidates are combined to make a new candidate.

Although he does not say so explicitly, Pass (1997) implies that the technology continuum is just one dimension of a 3-dimensional view that fosters solutions to problems that might not otherwise be achievable. It also encourages the integration of technologies. The three dimensions are technology, application, and the level of impact a technology has on solving a particular problem in the given area of application. This 3-dimensional view is illustrated in Figure 6.



**Figure 6. 3-Dimensional View of Adaptive Automation Technologies**

Figure 6 offers that a range of technologies are available to use for solving the range of problems posed by various applications. Depending on the type of problem and how much is known about it, a given technology will have more or less to contribute to a solution. Each dimension is continuous. It is possible that a given application will pose

several problems and that multiple technologies will need to be combined to provide a more meaningful solution than only one technology could by itself. This 3-dimensional view of adaptive systems technologies and their application broadens the scope of problems that can be solved and it opens the door to technologies that might not have been previously considered.

Additional support for this idea is given by Engels, Lindner, and Struder (1997), "An important success factor for the field of [adaptive automation] lies in the development and integration of methods ..." They go on to apply these ideas to the conduct of a project using steps similar to those discussed earlier. They end by coupling method selection with project subtasks. Thus, they tend also to lean toward a 3-dimensional view. An interesting point they bring out is that the solution to a given problem may require several subtasks, each of which may require the application of one or more technologies. So, while the view can be taken on a problem-by-problem basis, it helps to see the process of arrival at a problem's solution as a series of subtasks. Each subtask is achieved as an adaptive subprocess.

### Summary

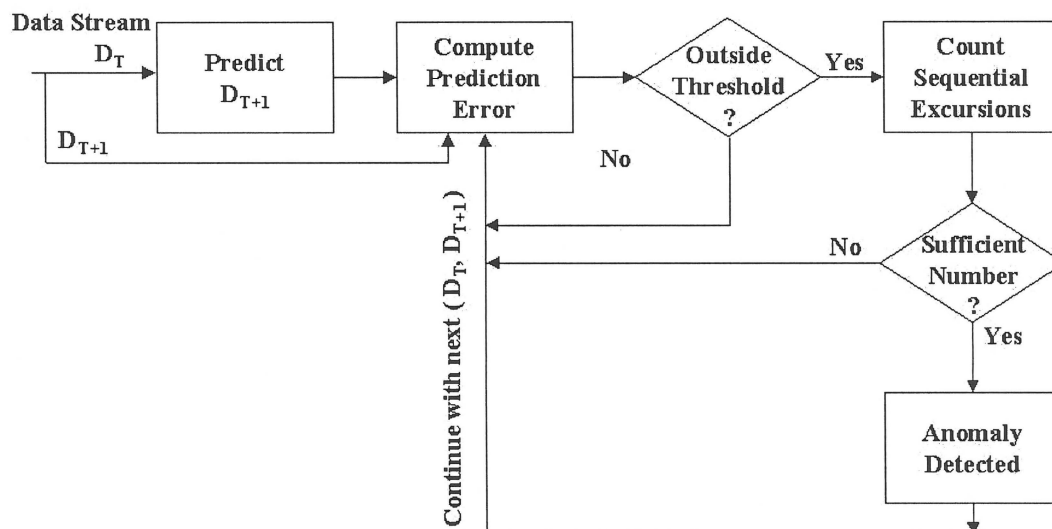
To avoid the traps of adaptive systems, a clear and objective process should be followed. This process should address not only the technical aspects of adaptive automation but also domain knowledge and bottom-line realities. Several technologies exist to support adaptive automation. These technologies exist along one axis of a 3-dimensional continuum that has the application and desired results along the other two axes. Combining an objective process with an expanded view of the technologies enables solutions that might not have previously been considered. In this way, the opportunities for producing meaningful results are greatly improved.

## Functional Experiments

There are three major components to the capability created by this research:

- prediction
- detection
- effective implementation for real-world problems

The innovation provided by this research is the employment of a prediction method from control theory to solve the problem of anomaly detection in data streams. How data stream next-sample prediction is used for detection is illustrated in Figure 7.



**Figure 7. Prediction as a Prelude to Detection**

Essentially, the predictor provides a difference measure between the actual and predicted samples. A detector decides if this difference is large enough often enough to postulate and confirm the existence of an anomaly. With the detection threshold and related processes taken in reverse, this approach also finds the end of confirmed anomalies.

The following subsections present the experimental design for the three components.

## Experiments in Prediction

Since this is not a dissertation on time-series prediction, the author has bypassed an in-depth discussion on that topic. Appendix A presents the technique chosen. It was selected because it has a firm grounding in well-established mathematics, it is readily implementable via parallel processing (course-grained and fine-grained), it is non-linear, and it has infinite support. (Infinite support exists in theory only since the fixed word-sizes of computers truncate very small values to zero.) For a brief introduction to time-series analysis see the author's two columns that overview that topic (Raeth, 12 Oct 01 and Raeth, 30 Oct 01).

Appendix A's next-sample prediction technique can be implemented in a language such as Fortran, C++, or Java. This technique is not language-dependent nor does the subsequent detection of anomalies depend on its peculiarities; except that, in the author's experience, non-linear methods tend to better represent the real-world than do linear methods, especially in highly dynamic situations.

### *Comparing the Gaussian Predictor with a Linear Predictor*

Although the author has had the experience that non-linear predictors' accuracy over linear predictors goes without saying, it is important that a demonstration be made.

Clearly, the heart of the adaptive detection threshold produced by this research is the ability to accurately predict the next value in a sampled data stream. While working with a natural and a synthetic data stream, the prediction ability of the present gaussian method was compared to that of a popular linear method in digital signal processing. The linear method performs a weighted sum of a number of the most recent past actual values. According to Morgan (1994, pp. 331-335), it is a very common method in digital signal processing. The summation is given below.

$$predictedValue = \sum_{i=1}^a w_i actual_i$$

Weights are adjusted via:

$$w_{i,t+1} = w_{i,t} + (2.0 * gain * (actual_{i,t+1} - predicted_{i,t}) * actual_{i,t+1})$$

where  $t$  is the current time

gain is a number indicating the rate of parameter evolution

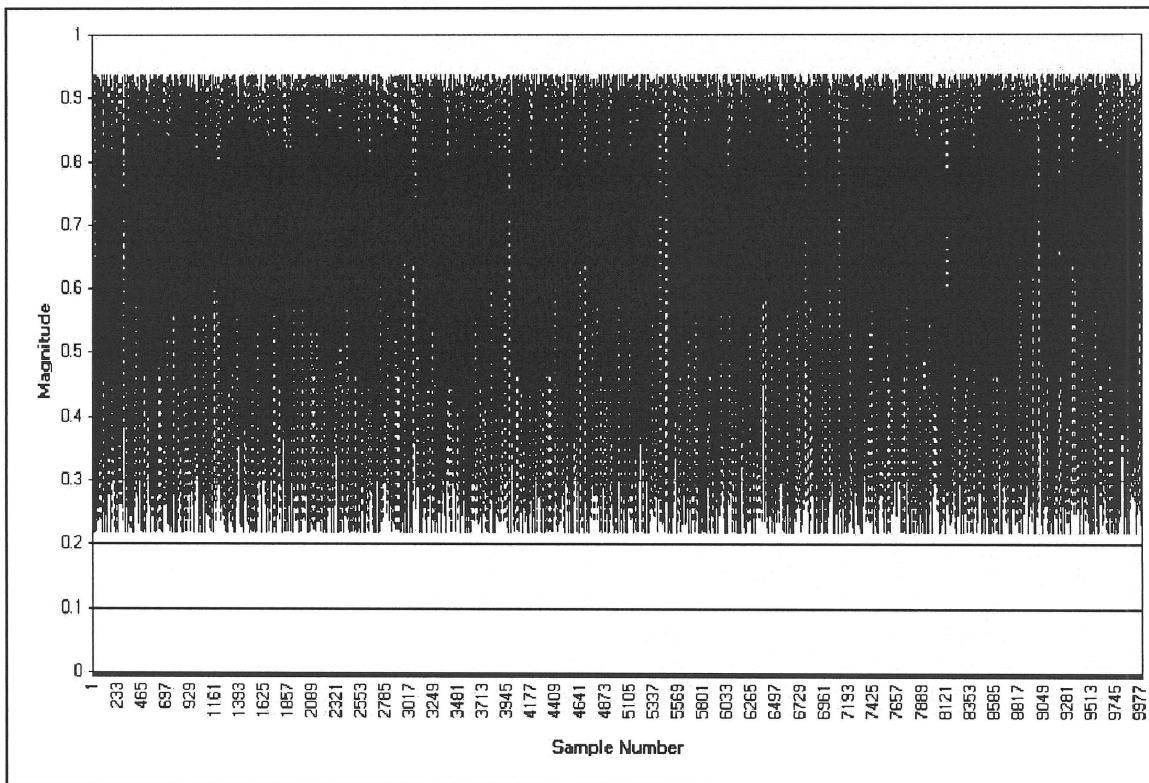
According to Morgan (1994, code comments), the past-value and weight buffers are initialized all at zero. The input has to be mapped to the range 0 .. 1 to ensure convergence.

The steps used for comparing the linear and non-linear predictors were:

1. Generate chaotic and natural input data
2. Run data through each prediction method
3. Generate four descriptive plots
  - magnitude of input data (Figure 8 and Figure 9)
  - range-relative prediction error (RRPE)
  - RRPE standard deviation
  - Fourier transform of input data  
This translates a signal from the magnitude to the frequency domain such that the original signal can be recreated via a summation of sinusoids of varying frequency and peak value. The spectral components are the frequencies of those sinusoids. The component values are the peaks of those sinusoids.
4. Calculate statistical significance of error and standard deviation differences.  
(See Appendix B for mechanization.)

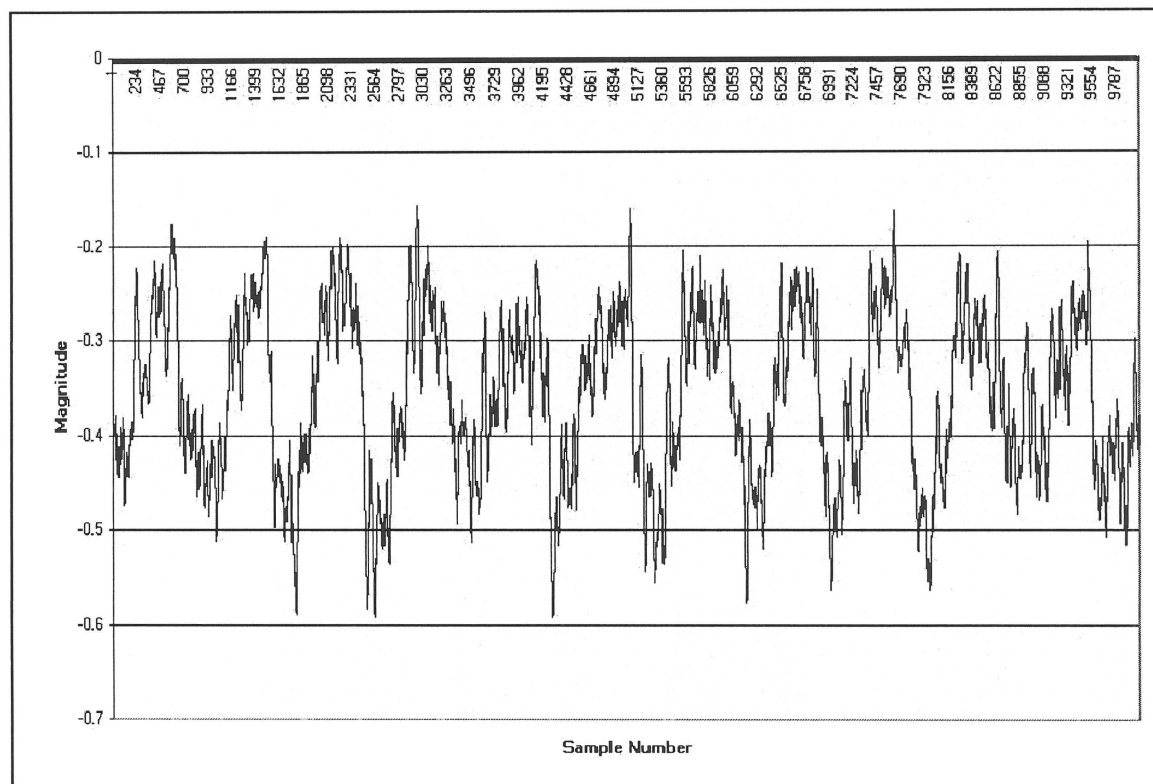
Figure 8 and Figure 9 show the chaotic and natural data magnitudes respectively. Chaotic data was generated by  $x_t = Cx_{t-1}(1.0 - x_{t-1})$  with  $C = 3.7513$  and  $x_0 = 0.9270$ . According to Hilborn (1994, p 26), this equation is chaotic when  $3.6 \leq C < 4.0$  and

$0.0 < x_0 < 1.0$ . Natural data was collected by mounting a microphone inside the frame of a common household box fan and sampling at 44100hz.



**Figure 8. Chaos original signal magnitude**





**Figure 9. Fan noise original signal magnitude**

Prediction error is defined to be  $P_E = | \text{Actual} - \text{Predicted} |$ . Range-relative prediction error is computed by dividing  $P_E$  by the size of the expected numeric range of the actual data,  $RRPE = P_E / (\text{Range\_Max} - \text{Range\_Min})$ . The average is calculated over  $N$  samples. This particular calculation relates the size of the prediction error to the actual data range.

RRPE is derived from a discussion by Masters (1993, pp. 64-66). He includes a calculation for Root Mean Square Error (RMSE).

$$RMSE = \sqrt{(1/n) \sum_{i=1}^n (\text{actual}_i - \text{predicted}_i)^2}$$

At issue with this equation, according to Masters (1993), is that it is sensitive to the range of the actual sampled values. To cancel this effect, this research divides the prediction error by the range of actual values.

The gaussian predictor was documented in the following table:

Gaussian Predictor Setup	
Gaussian Width	
Number of Terms	
Gain Numerator	
Number of Samples	
Data Stream Name	

The linear predictor was documented in the following table:

Linear Predictor Setup	
Number of Terms	
Gain	
Number of Samples	
Data Stream Name	

RRPE results were compared using the statistical significance table and procedure discussed in Appendix B.

### *Single-Dimensioned Data Streams*

A series of experiments were performed with the Gaussian predictor applied to various single-dimensioned data streams. There are two user-settable parameters:

- gain numerator, G
- number of gaussian terms, N

For each data stream, these parameters were varied according to the following table:

Results Table: name of signal						
Gain Numerator						
Number Of Gaussian Terms		0.01	0.10	0.50	1.00	1.50
	2					
	20					
	200					
	2000					
	20000					

Intersecting cells contain the overall average of the range-relative prediction error. Since the purpose of the prediction trials is to establish that the next-sample can be estimated to an acceptable level of accuracy in various situations, the table was exercised on 2000 samples of the following data streams.

- Constant Value = 1.0
- Sinusoid =  $(\sin(\text{sample\_number}/20.0) / 3.0) + 0.34$
- $\text{Chaos}_{t+1} = 3.7513 * \text{Chaos}_t * (1.0 - \text{Chaos}_t)$ ,  
 $\text{Chaos}_0 = 0.9270$ , see Figure 10.
- Noise recorded by microphone from a box fan, see Figure 11.

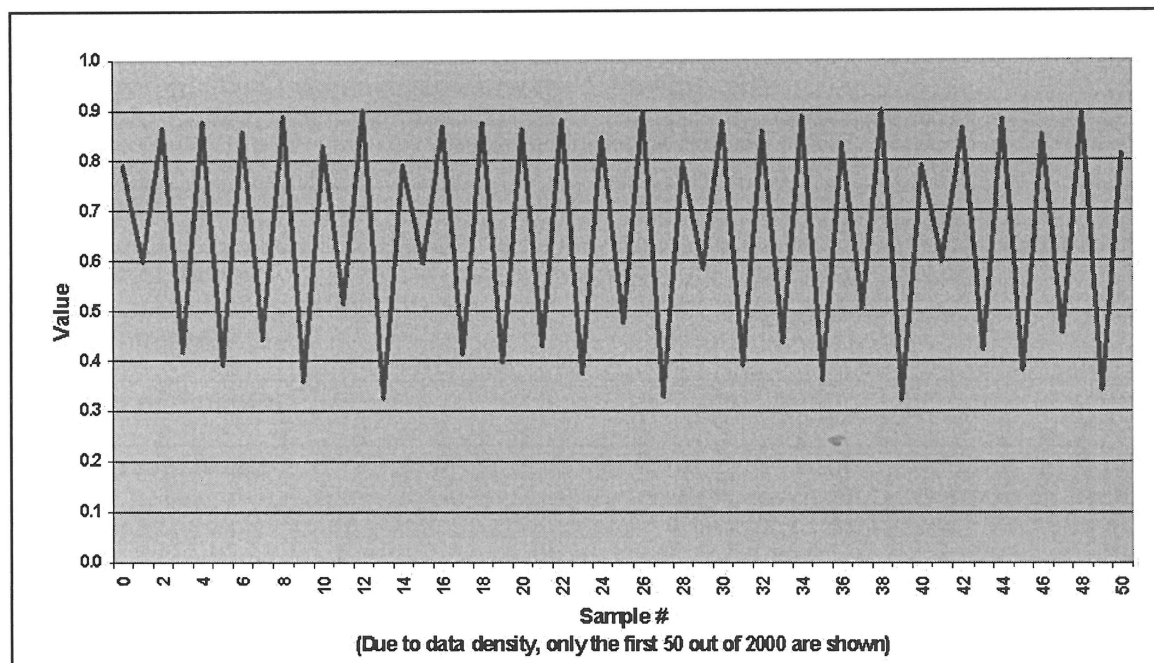


Figure 10. Illustration of Chaotic Data

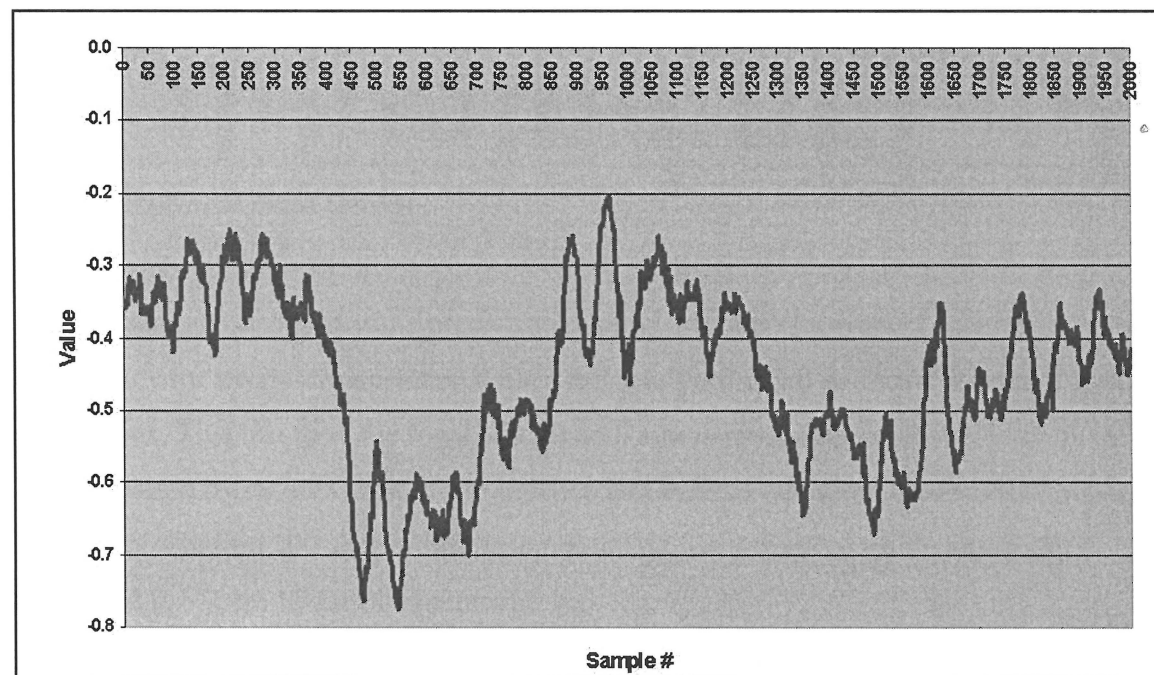


Figure 11. Illustration of Fan-Noise Data

Each experiment was documented using the following table:

Gaussian Predictor Setup	
Gaussian Width	
Number of Terms	
Gain Numerator	
Number of Samples	
Data Stream Name	

Best and worst RRPE curves from each set of experiments with each data stream were compared using the statistical significance table and procedure described in Appendix B.

This test had the value of observing the ability of the prediction method to foretell the next sample in various data streams and of providing a sensitivity analysis relative to the user settable parameters.

### *Two-Dimensional Data Stream*

Once it was established that the prediction method could perform to an acceptable level of accuracy for single-dimensioned data, a test was performed on a data stream of two dimensions. This test took the form of a 10 x 10 simulated frame sequence. Each pixel was generated by its own unique and independent chaotic equation. Essentially, this came down to combining 100 single-dimension tests that use variations on the equation illustrated in Figure 10 for one parameter set.

The chaotic equation employed in the one- and two-dimensional prediction tests is  $x_t = Cx_{t-1}(1.0 - x_{t-1})$ . According to Hilborn (1994, p 26), this equation is chaotic when  $3.6 \leq C < 4.0$  and  $0.0 < x_0 < 1.0$ . For this two-dimensional test,  $C$  and  $x_0$  were varied by random selection from a uniform distribution over their ranges. While a frame generated

in this way can be color-coded, it makes an uninformative image. Therefore, a figure is not provided. The report on this test plotted range-relative prediction error averaged across all pixels on the Y axis with frame number on the X axis. It also plotted the evolving standard deviation of the average RRPE. The model setup was documented using the following table. All pixel models had the same setup.

Two-Dimensional Frame Prediction Experiment Setup	
Gaussian Width	
Number of Gaussian Terms	
Gain Numerator	
Total Frames in Experiment	

At this point it is worth reckoning with the phrase “acceptable prediction accuracy”. The detection method to be described later adjusts its threshold based on the prediction accuracy. Therefore, there is no domain-independent definition of “acceptable prediction accuracy”. Much depends on how different the anomalies are expected to be from the background. If the anomalies exceed the background by only 10%, for instance, the prediction error would have to be less than 10%. However, the author’s feeling is that anything beyond 20% error would be too domain limited. The intent of the research is to arrive at an independent capability that makes only loose assumptions on the domain.

### Experiments with Detection

Detection is actually quite simple, being based on elementary statistics:

1. Range-relative prediction error (RRPE) is averaged over a window of samples. An average of the actual value is also maintained over the same window. In this step we are creating two moving averages.
2. Based on a knowledge of the domain, an RRPE threshold is chosen beyond which the immediate (local) RRPE is said to depart from the acceptable when compared to the average RRPE. When such a departure occurs, the beginning of an anomaly is postulated. The threshold is defined in terms of a fraction of

the average RRPE. For instance, one might say that a 10% departure from the average is an indication that an anomaly might be starting.

3. A count is kept of the number of times in a row the RRPE departure originally appears. If, during the count, the RRPE fails to exceed its threshold, the count is reset.
4. Based on a knowledge of the domain, a count threshold is chosen beyond which the departure count confirms the beginning of an anomaly. This prevents outliers from being confirmed as anomalies.
5. At this point, the prediction model stops evolving so that the anomaly does not become part of the background. Instead, the prediction model simply outputs the average of the window's actual values collected prior to the first sample of the anomaly. This is because the predictor is not a model of the data stream, just a next-sample forecaster. An alternative to the average is to output as the prediction the last actual value prior to the first sample of the anomaly. The RRPE average is also fixed upon reaching this step.
6. RRPE continues to be collected while the anomaly is in being. However, the predictor is simply producing the value determined in the previous step.
7. Based on a knowledge of the domain, an RRPE threshold is chosen such that the end of the anomaly is postulated when the RRPE falls below that threshold. Here, the threshold is stated as a fraction of the average RRPE previously computed prior to the start of the anomaly.
8. A count is kept of the number of sequential times the local RRPE falls below the threshold. If, during the count, the RRPE does not fall below its threshold, the count is reset.

9. Based on a knowledge of the domain, a count threshold is chosen such that the end of the anomaly is confirmed when the count from the previous step exceeds that threshold.

How to measure prediction and detection success is application dependent. An example is interceptor vectoring via object trajectory prediction (Raeth, 1999, Jul). Even though the interceptor may frequently make better than 100% error in predicting the object's trajectory, the interceptor still can come in physical contact with the object, certainly within the interceptor's activation footprint. So, simply measuring prediction accuracy is insufficient. The larger question involves the goals of the data collection and processing.

In the case of an image stream, we want to identify those pixels where something besides the background exists. Success would be measured in terms of how accurately we can place the start-time and end-time markers for each event. We would also like to blank out those portions of the display that contain the background and only show those portions that contain possible anomalies. In the case of this project, we are interested in the number of false alarms, good detections, and missed detections. Even here it is necessary to establish just what false alarms and missed detects are. For instance: Consider an object that moves through a scene and thereby changes the lighting conditions. It is possible that this change will cause the model to postulate an anomaly based only on the pixels affected by the lighting change. Is this a false alarm? Since the model has not detected the actual object, is this a missed detect? The model has not detected the object itself but has detected evidence of its presence. This research did not count the lighting change detects as false alarms but did count the missed object as a missed detect.

Another question is how long it takes to confirm the presence of the anomaly. Is it a missed detect if the anomaly exists for 10 samples but is only confirmed for the last 5? Is it a false alarm if it takes 4 samples to confirm that the anomaly no longer exists? It depends on the detail of the analysis. In the end, user intent has to drive the analysis. It may be more important to notice that the anomaly exists at all and to tell where it is located, in the temporal and spatial sense. In that sense, if the anomaly is confirmed to



exist while it is still in being, the model could be said to have achieved a good detect. Ultimately, there is the matter of detecting the anomaly soon enough to support a meaningful and timely reaction. If the anomaly exists for 10 samples, does a confirmation covering the last 5 samples give sufficient reaction time? In the case of this research, detecting the anomaly while it is in being was considered a good detect for that anomaly. A measure of the length of time the anomaly was actually detected was given relative to its total existence. There will also be a measure of the length of time taken to determine that the anomaly no longer exists. This latter measure gives a sense of the length of the anomaly's "wake", the length of time the anomaly is thought to still exist when it no longer exists.

For data streams generated from equations, it is possible to exactly specify at what sample an anomaly is to start and at what sample an anomaly is to stop. In a frame sense, it is possible to specify exactly what pixels will contain what anomalies when and for how long. For natural data, such precise specification is not possible given the available equipment. Thus, in the case of natural data streams, an estimate of false alarms, missed detects, and good detects was made. Figures give the reader a sense of the validity of these estimates. Plots of detection data were also produced (# detects per frame).

### *Detection in Single-Dimensioned Data Streams*

To begin the detection experiments with single-dimensioned data streams, the comparison between linear and gaussian methods were continued to include a detection experiment. This experiment used the previous fan noise with a loud, obvious, "Hello" electronically mixed into the signal. The resulting signal is illustrated in Figure 12. A comparison of the spectral plots is given in Figure 13.

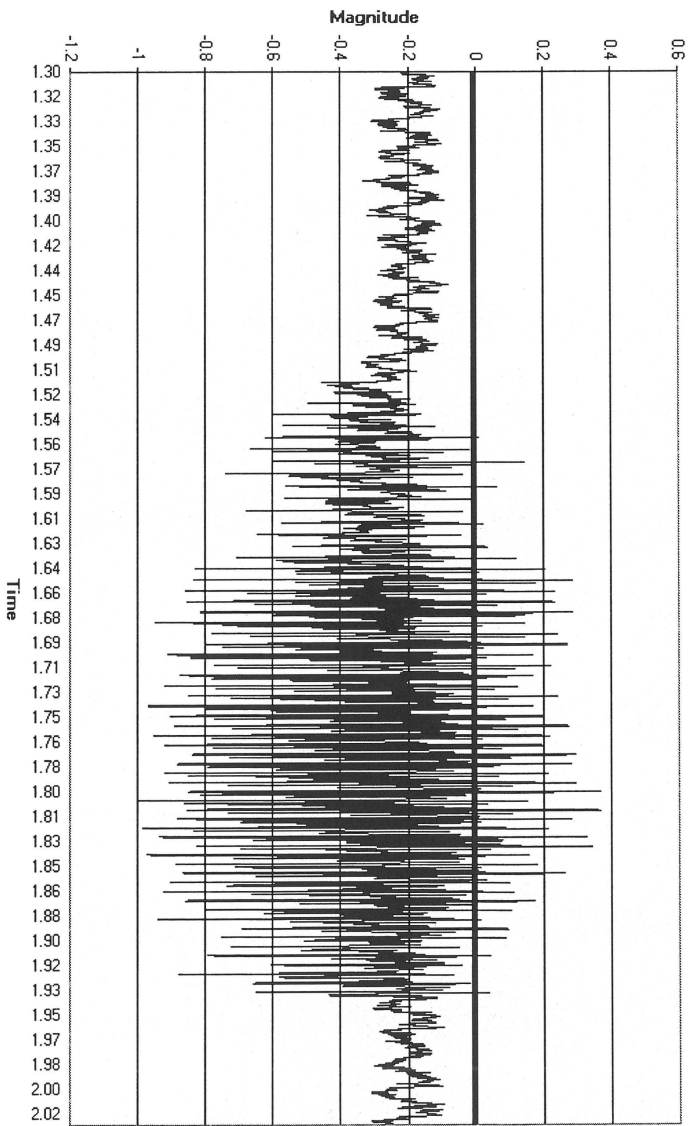


Figure 12. Fan noise with loud "Hello" mixed in

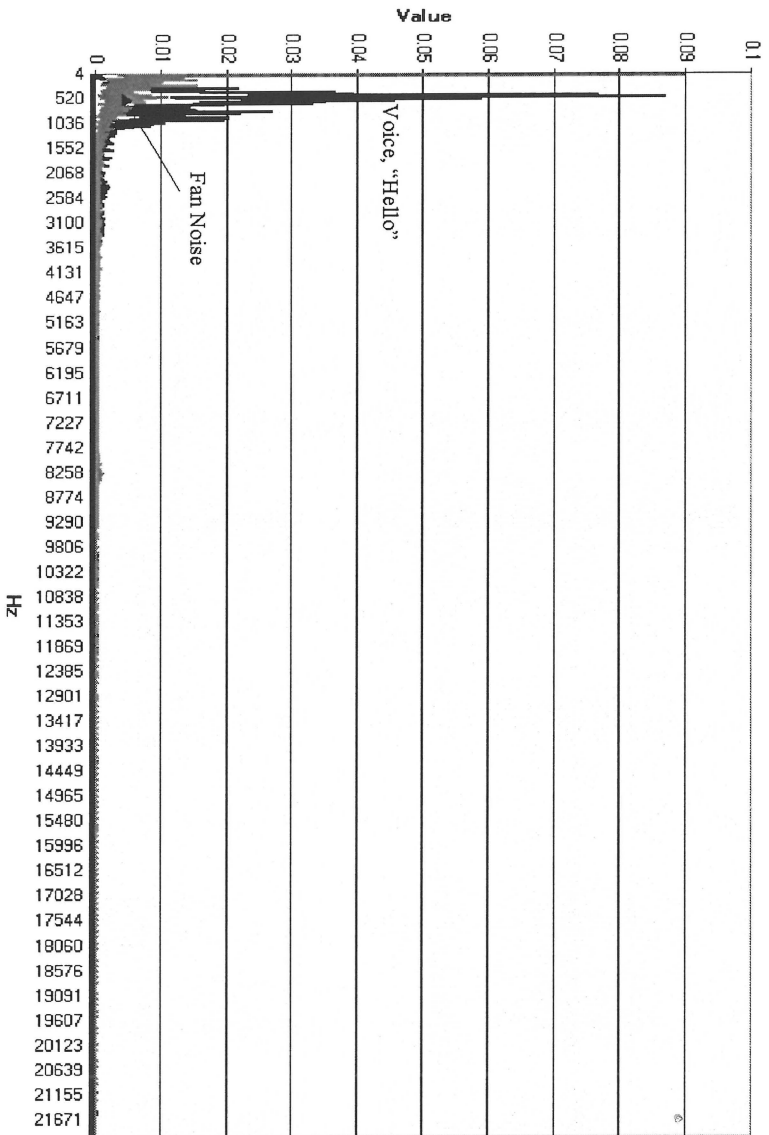


Figure 13. Spectral plot comparing fan noise and the word "Hello"

The following tables described the experiment's setup:

Gaussian Predictor Setup	
Gaussian Width	
Number of Terms	
Gain Numerator	
Moving Average Length	

Linear Predictor Setup	
Number of Terms	
Gain	

Detector Setup for both the Gaussian and Linear Models	
# samples required to confirm anomaly start	
Threshold indicating an anomalous sample	
# samples required to confirm anomaly end	
Threshold indicating a non-anomalous sample	
Data Stream Name	
# samples in data stream	

A combination of synthetic and natural single-dimensioned data streams was used to perform additional detection tests with the gaussian model:

- Constant Value = 5.0, with square pulse = 10.0 inserted at  $t = 300$  and remaining through  $t = 400$ .
- Sinusoid =  $(\sin(\text{Time}/20.0) / 3.0) + 0.6666$ , with  
Cosine =  $(\cos(\text{Time}/20.0) / 3.0) + 0.6666$  inserted at  $t = 1000$  and remaining through  $t = 1500$ .
- Chaos =  $3.7513 * \text{ChaoticResult}_t * (1.0 - \text{ChaoticResult}_t)$ ,  
 $\text{ChaoticResult}_t$  at  $t_0 = 0.9270$ , with square pulse = 0.5 inserted at  $t = 600$  and remaining through  $t = 650$ .

- Noise recorded by microphone from a box fan, with reduced-volume “Hello” recorded by the same microphone inserted via electronic mixing. (Available equipment does not permit exact insertion at a given sample range.)

The setup for each experiment was documented using the following table:

Single-Dimensioned Data Stream Experiment Setup	
Gaussian Width	
Number Gaussian Terms	
Gain Numerator	
# Sequential Unpredicted Samples Required for an Anomaly to be Confirmed	
% Difference Required Between Predicted and Actual for a Sample to be Considered Unpredicted	
# Samples Before Anomaly Was Inserted	
# Samples Before Model Entered Detection Mode	
Total Samples in Experiment	

As the constant-valued data stream is expected to deliver the best detection results, the RRPE curve from that experiment was compared to each of the other three experiments using the statistical significance table and procedure from Appendix B. Plots comparing the original and filtered data streams will also be provided. ©

### *Detection in Two-Dimensional Data Streams*

One can see from the prediction mathematics described in Appendix A and the detection procedure described above that this technology is designed for a single-dimensioned data stream. Even so, via multiple applications of the technology, n-dimensional problems can be addressed. Each element in the n-space has its own time-varying single-dimensioned data stream. Commonly, for imagery, the two-dimensional elements are called pixels. To apply this technology, each pixel has its own prediction/detection model. Each model is built automatically from sequential observations of its pixel's data stream, without reference to any other pixel. As the model is being built, it gradually becomes able to predict the next pixel value in the sequence. Once construction is completed, the model continues to evolve as new samples arrive.

Pixels with a confirmed anomaly stop evolving until the anomaly is no longer present. The other pixels keep evolving. In this way, in keeping with the detection procedure, the anomaly does not become part of the pixel's prediction. At the macro level, the image anomaly does not become part of the next-frame prediction, thus preserving a sense of how the non-anomaly frame should look.

Experiments conducted to this point were designed to demonstrate this technology's underlying abilities. Subsequent experiments applied these abilities to image streams.

- Two-dimensional chaotic-equation prediction experiment with insertion of a value = 0.5, 10-sample, pulse moving along the lower-right to upper-left diagonal. This is achieved by inserting the pulse into the subsequent pixel at the end of the 10-sample sequence. In this way, a moving anomaly is simulated.
- 10-by-10 low-resolution image generated by a proprietary infrared camera simulator. An anomaly moving from the top of the frame to the bottom, in the same column, was inserted into the right side of the image.
- QuickCam camera set to low picture sharpness and 160 x 120 resolution aimed at a scene that does not change at all. A model car was pulled by string through the scene. The RGB triplet produced by the camera for each pixel was combined to produce a single value by adding the individual triplet elements. The goal was to filter out all but the car.
- Under the same setup, the QuickCam aimed at a highly dynamic scene. The scene was produced by pointing the camera at a ceiling fan having a room light mounted under the fan housing. The fan was turning at such a speed that the camera undersampled the scene. This caused the fan blades to appear to be constantly switching directions. A model airplane was pulled by string through the scene. (Note that the scene was sampled below its Nyquist rate. A

characteristic of the QuickCam is that a fixed sampling rate can not be set. The camera lens opens and stays open until sufficient light is received to record a frame. Thus, it was necessary to experiment a bit with the fan speed and lighting conditions to achieve the specified dynamic scene. More expensive equipment would make this process more precise but such equipment was not available to this project.)

The following table was used to document the setup of each experiment:

Two-Dimensional Data Stream Experiment Setup	
Gaussian Width	
Number Gaussian Terms	
Gain Numerator	
# Sequential Unpredicted Frames Required for an Anomaly to be Confirmed	
% Difference Required Between Predicted and Actual for a Pixel to be Considered Unpredicted	
# Frames Before Object Entered Scene	
# Frames Before Model Entered Detection Mode	
Total Frames in Experiment	

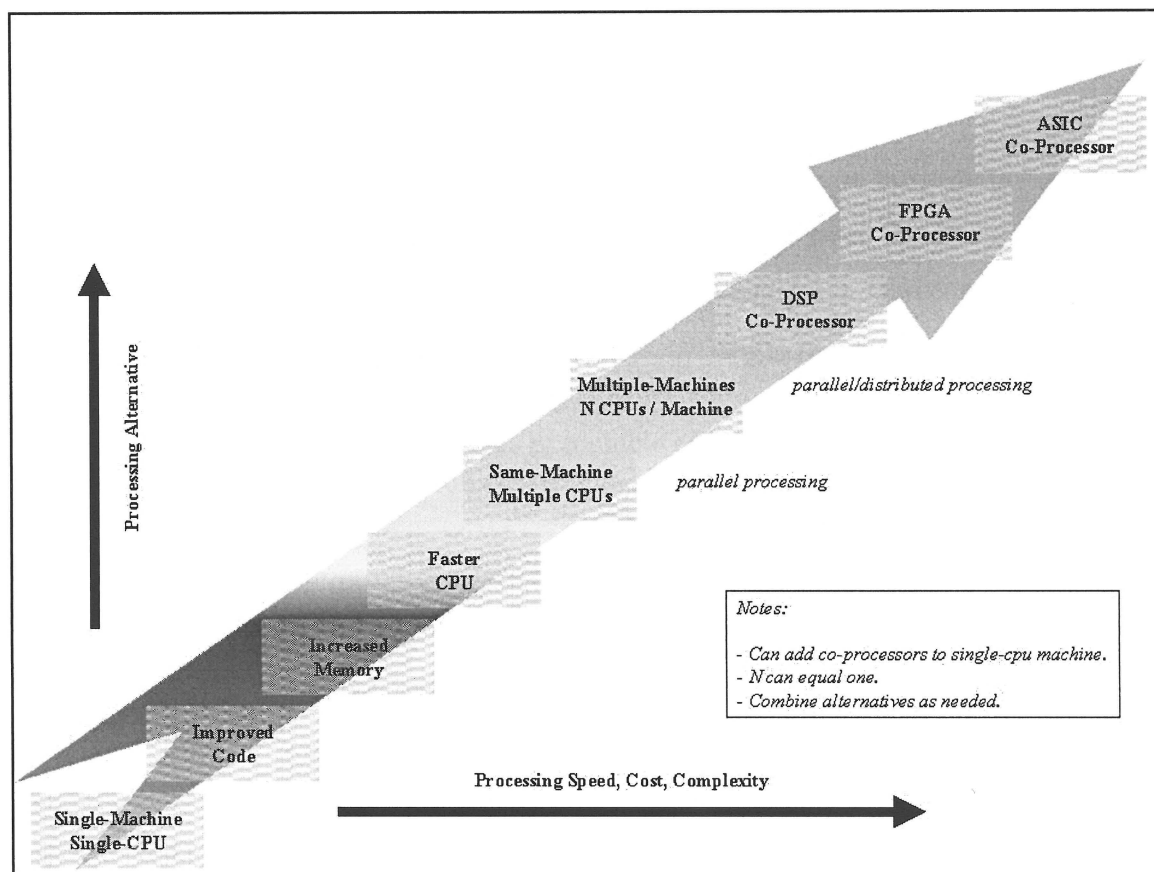
In the case of the chaotic image stream, plots of the number of false alarms and good detects were produced. For the QuickCam and simulated infrared image streams, false alarms and good detects were estimated since, given available equipment, it was not possible to exactly place the anomalies in the scene. In all cases, screen shots showing a sample of the filtered image were generated.

### Implementation Experiments

After nearly 30 years in the computer engineering profession, it has been the author's experience that there is a latent demand for throughput that always exceeds what present-day hardware and software can deliver. The demand is "latent" because users of technology are convinced that it is necessary to "wait for the next round of hardware", that technology is evolving so fast that the speed will be there within the next year or so. Yet, when that year passes and the needs of a year past are satisfied, the present-day need has again increased beyond what present-day technology is able to satisfy. Thus, although

user's do not demand the throughput because they are convinced it is not available, the need is there none-the-less. The author is persuaded that it is not necessary to wait for the next round of technology in order to satisfy the present-day need for throughput. Rather, it is possible to satisfy present-day needs with present-day technology by employing technology in ways that are not traditional. It is not necessary to view computing through the limited scope of the traditional office computer or engineering workstation. That is why this section is included in this dissertation. The intent is to show that the prediction/detection technology described here is implementable for real-world applications. Another purpose is to explore ways of using present technology to satisfy present needs for throughput, while laying the groundwork for satisfying future needs. In this way, it is possible move applications at least one year ahead of the technology curve, rather than always leaving them one year behind. Therein lies a competitive advantage.

There were four phases to the throughput experiments. These were designed to push the technology produced by this research toward implementation for real-world problems and to show that the anomaly detection technique developed in this research is not just for toy problems. Figure 14 shows that there are many steps that could have been taken.



**Figure 14. Throughput improvement alternatives**

The first phase improved the code that was running on an architecture with a single machine containing one CPU. Then experiments using an architecture with a single machine and multiple CPUs were conducted. Following those experiments were efforts with an architecture having two networked machines, each with four CPUs. Finally, a thrust was taken at converting the technology for implementation on Field-Programmable Gate Arrays. This last phase simply examined the difficulties and was not intended to achieve actual implementation. The purpose was see what it would take to step out of the box of fetch/execute computing methods.



### *Single-Machine Single-CPU Throughput Experiment*

In this experiment, work was done on the original code to increase the throughput of a single-CPU computer. The intent was to build the model to a given level of accuracy in less clock time. Image size and observation sequence remained the same throughout each trial. The changes in the code and its method of construction included:

- method of compiling the source code
- method of preprocessing the executable code
- number and construction of statements used to implement the model

Other opportunities to improve the base code would be to write certain compute-intensive routines in assembler and to perform other machine-specific improvements. In order to keep the code as generic as possible, these approaches were not explored.

The first trial in the experiment repeated the simulated IR-camera experiment after it was instrumented to collect per-cycle timing data. The second trial applied all improvements. A statistical significance table and procedure described in Appendix B was used to compare the per-cycle timing data from the two trials. (“cycles” are the processing performed due to the arrival of each sample.) This experiment produced more efficient code for use by later throughput experiments.

### *Multi-CPU Throughput Experiments*

As a basis of test planning for further throughput experiments, an objective measure of execution speedup was needed. A subjective approach leads too easily to incorrect findings. Bailey lists twelve ways to produce glowing but false reports from evaluations of parallel-processor configurations:

1. Use 32-bit arithmetic, not 64-bit arithmetic, for performance benchmarks. Compare 32-bit performance to 64-bit performance.

2. Assume that the inner kernel of an application is the sole determinant of performance.
3. Use assembly code and other low-level language constructs for performance and compare them with Fortran or C implementations.
4. Scale the problem size with the number of processors, but do not disclose this fact.
5. Estimate linear scaling of performance without proof.
6. Compare the performance of heavily optimized benchmarks against unoptimized benchmarks.
7. Compare with an old code on an obsolete system.
8. Base MFLOPS operation counts on the parallel implementation instead of on the best sequential implementation.
9. Give performance in terms of processor utilization, parallel speedup, or peak MFLOPS per \$.
10. Use algorithms that are numerically inefficient, compared to the best known serial or vector algorithms for an application, in order to show artificially high MFLOPS rates.
11. Measure parallel run times on a dedicated system, but measure conventional run times on a heavily loaded system.
12. If all else fails, show pretty pictures and animated videos, and don't talk about performance.

To avoid subjectivity, this project used the following metrics:

- Execution time was collected and reported for one through  $n$  processors on a shared-memory bus. Later, this was increased by one processor up through  $m$  processors, with  $(m - n)$  processors being on their own separate memory bus and linked to the application via a network and message-passing infrastructure. The resulting table of execution times was used to estimate the value of '  $f$  ' in Amdahl's Law for a particular implementation of the model. See Appendix C for an explanation of this parameter.

- According to Morse (1994, p 33), the actual speed-up is the time it takes to run the code using one processor divided by the time it takes to run on  $N$  processors. The ideal speed-up is equal to the number of processors. Efficiency is the actual speed-up divided by the ideal speed-up. This can also be calculated as the actual speed-up on  $N$  processors divided by  $N$ . Speed-Up and Efficiency was collected and reported. However, do not confuse the two. With increasing speed-up, it may be possible to meet a throughput goal. In doing so, it may be necessary to add a number of processors such that efficiency drops.
- Summary charts compare the values collected and clearly show the point at which adding more processors is not cost effective.
- During the experiments, the code and model size were not changed. The comparisons were of the exact same code and the exact same model running on increasing numbers of processors.

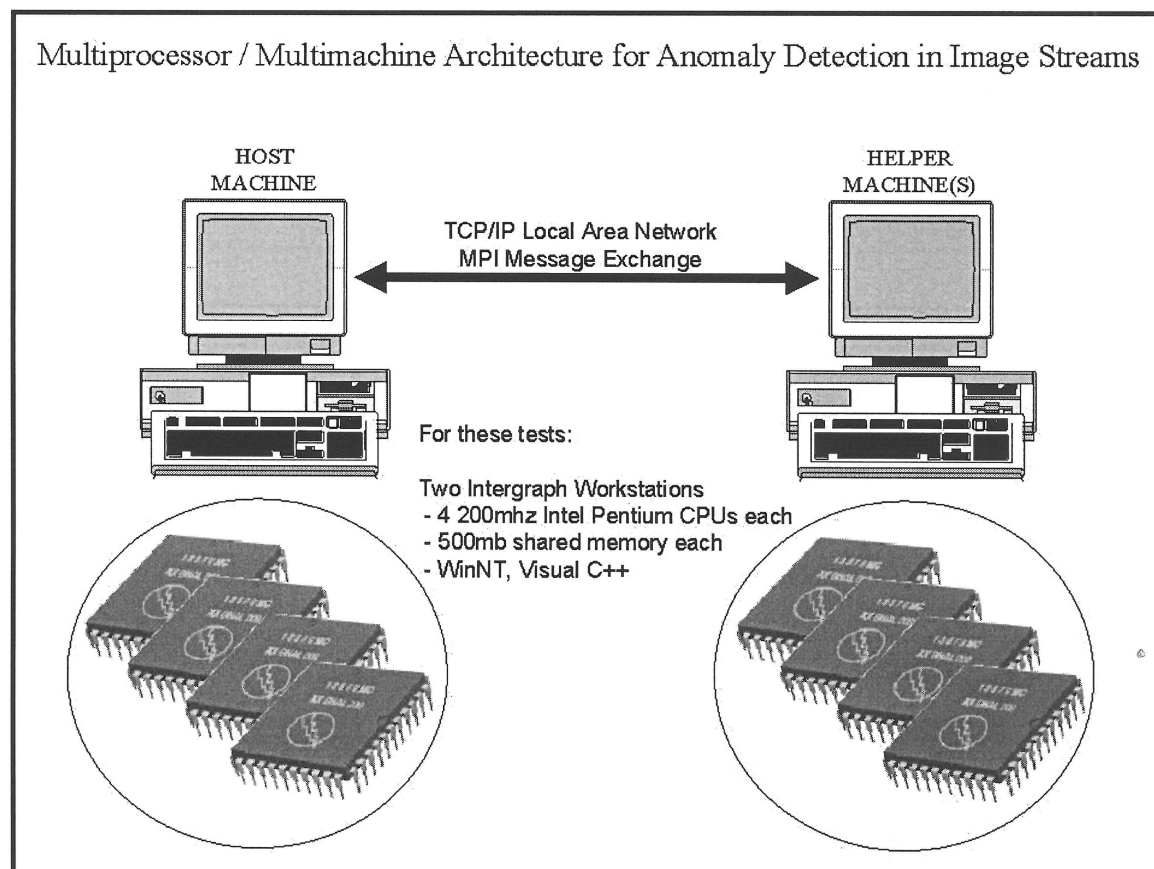
Each run increased the number of processors employed on a single machine. A multi-CPU version of the model was created to support this experiment. It was considered likely that this version of the model would have additional overhead for the use of  $N$  processors even if  $N = 1$ .

### *Multi-Machine Multi-CPU Experiments*

A third version of the architecture was generated to accommodate the multi-machine / multi-CPU case. This new version was not hardwired. Rather, it automatically expanded itself according to the number of machines, the number of CPUs on each machine, and the capacity of each machine known to be dedicated to the model's workload. Although this third version was capable of running even the single-machine / single-CPU case, it was expected that the overhead would increase because of the new code based on  $M$  machines instead of assuming just one. This is similar to the previous expectation when

the single-machine /single-CPU version was expanded to single-machine / multi-CPU. Removing the assumption of just one CPU is expected to add some overhead to the processing.

The physical computer arrangement used is illustrated in Figure 15.



**Figure 15. Computer physical arrangement for multi-machine/multi-CPU tests**

### Experiment #1A

This experiment ran the model with 8290 frames having a resolution of 160 x 120 pixels. (A pixel is represented by 4 bytes.) Since the prediction model for each pixel had 30 terms, the total number of terms per frame is  $160 \times 120 \times 30 = 576,000$ . (There are some applications where this size image is useful.) The modeling software for this experiment

was designed to support only one machine and one processor. The run-time was collected for comparison with the 1-processor case in Experiments #1B and #1C.

#### Experiment #1B

This experiment employed code designed for 1 through P processors on a single machine. The run time with 1 through 4 processors was collected. The speed-up and efficiency was calculated. Data was collected in the following chart. Plots of run time, speed-up, and efficiency were generated Vs number of processors.

# CPUs	Run Time	Speed Up	Efficiency
1			
2			
3			
4			

#### Experiment #1C

This experiment is the same as #1B except that the code can support multiple machines. The caveat is that the code was restricted to run on only one machine. A message-passing infrastructure was brought up and the was code run from the cluster control interface.

#### Experiment #2

This experiment continued experiment #1C by adding 5 through 8 processors to the mix via cluster computing. As more processors were added from the additional machine, a larger image chip was sent for that machine to work on. Each machine had a workload capacity number assigned to it. This number was equal to the amount of time the machine took to do a sample image stream problem in a stand-alone setting using all of its processors. Thus, the larger the number, the lower the workload capacity. The total cluster capacity is the sum of all the numbers from the various machines. The size of the

image chip sent to a given machine is calculated as a fraction of that machine's share of the total cluster capacity:

$(\text{total\_image\_size} * \text{machineX\_cluster\_capacity\_fraction})$ . The fraction is equal to  $(1 - (\text{machineX\_capacity} / \text{total\_cluster\_capacity}))$ .

### Experiment #3

This experiment used software designed to only measure the overhead inherent in the model. There were four phases:

- A. Read the frames from a local file.
- B. Send a 4-bytes-per-pixel half-sized frame chip on to another machine and receive a 1-byte-per-pixel chip back.
- C. Store the frames on local disk.
- D. Display two copies of the frame on the CRT (original and filtered).

The following table was used to estimate the impact of all four activities on the run times in experiments 1A, 1B, and 1C. (Results from experiment 3B are not applicable to experiment 1A or 1B since they do not send chips to other machines. Similarly, experiment 3B results are not applicable to the single machine case in experiment 1C.)

# CPUs	Run Time	A. % Due to Frame Receive	B. % Due to Inter- Machine Communication	C. % Due to Frame Storage	D. % Due to Frame Display
1					
2					
3					
4					

The following table collected the underlying data.

Overhead Activity	Total Processing Time	Avg Time Per Pixel
A. Receiving all Frames		
B. Send / Receive Comm with one other Machine		
C. Storing Frames on Local Disk		
D. Displaying two Frames on Screen		

### *Exploring Issues In Moving Toward FPGA Implementation*

This part of the research continued the exploration into alternative means of improving throughput. The task was to investigate a means of translating the computationally intensive parts of the model for use by Field Programmable Gate Arrays (FPGAs). The track involved the following steps:

1. translating part of PAD to Java
2. ensuring that part of the model executes correctly in a standard Java environment
3. explore the idea of using a commercial tool (Forge) to translate the Java byte codes to Verilog, a hardware description language
4. actively participating on the Xilinx' Forge beta test team

This task was supported by contacts at Xilinx and gaining a spot on the Forge beta test team. The thrust here was to demonstrate a smooth flow between software applications and high-speed signal processing hardware without having to program directly in assembler or a hardware description language. Forge is unique in the market in that it is intended to take object-oriented source code written using a standard application development environment, and translate it directly for FPGA use without having to modify the programming language with special keywords or other functionality. This

## Chapter Four

### Results

One of the points of the author's work is to produce a technology that is fit for more than toy problems. Therefore, synthetic data was used for initial testing and data from actual sensors was used for additional testing. Further, an exploration was made of various means of physical implementation. So, the results presented here are from applications of the theory and from practical realization.

#### Results From Applying the Theory

This chapter on Results mirrors Chapter 3 on Methodology. To that end, the same section titles are repeated. Each section starts with the conclusions. These are followed by the supporting data.

#### Experiments in Prediction

##### *Comparing the Gaussian Predictor with a Linear Predictor*

For the data tried, the original assertion was confirmed. The Gaussian (non-linear) predictor proved to be superior to the linear predictor in terms of accuracy. In all experiments, a statistically significant difference in results was observed.

Since Morgan (1994, pp. 331-335) asserts in his code comments that the data has to be mapped to the range 0 .. 1 for the linear case, the linear and Gaussian models were exercised on mapped and unmapped data in both the Chaotic and FanNoise experiments. The following tables show how the Gaussian and linear models were arranged:



Gaussian Predictor Setup	
Gaussian Width	11.28125, per calculation described in Appendix A
Number of Terms	20, centered evenly over the data max/min range +/- 0.5
Gain Numerator	0.5, based on general experience
Number of Samples	32000, maximum number of points that can be plotted by Excel spreadsheet
Data Stream Name	Chaos and FanNoise

Linear Predictor Setup	
Number of Terms	250, per Morgan's choice for his noise experiment
Gain	0.0001, Morgan chose a higher value for his noise experiment but that value did not work well for these experiments
Number of Samples	32000
Data Stream Name	Chaos and FanNoise

If it is assumed that the chaotic signal was sampled at 44,100hz then there were meaningful frequency components through 20,000hz, as shown in Figure 16. Given its 44,100hz sampling rate, the data derived from the fan noise had meaningful frequency components through 1,400hz as illustrated in Figure 17. The spectral plots were calculated over 16384 datapoints, the maximum allowed by the student version of the DADISP engineering spreadsheet.

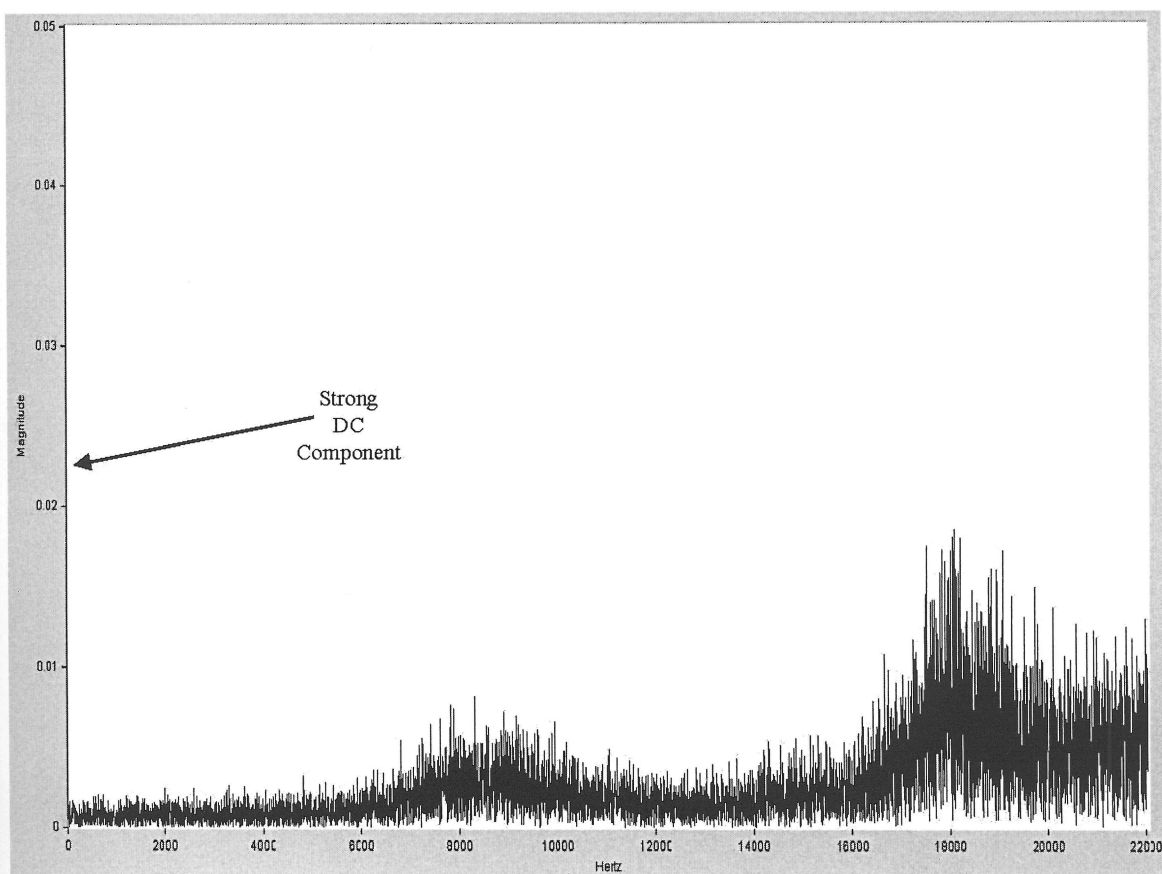
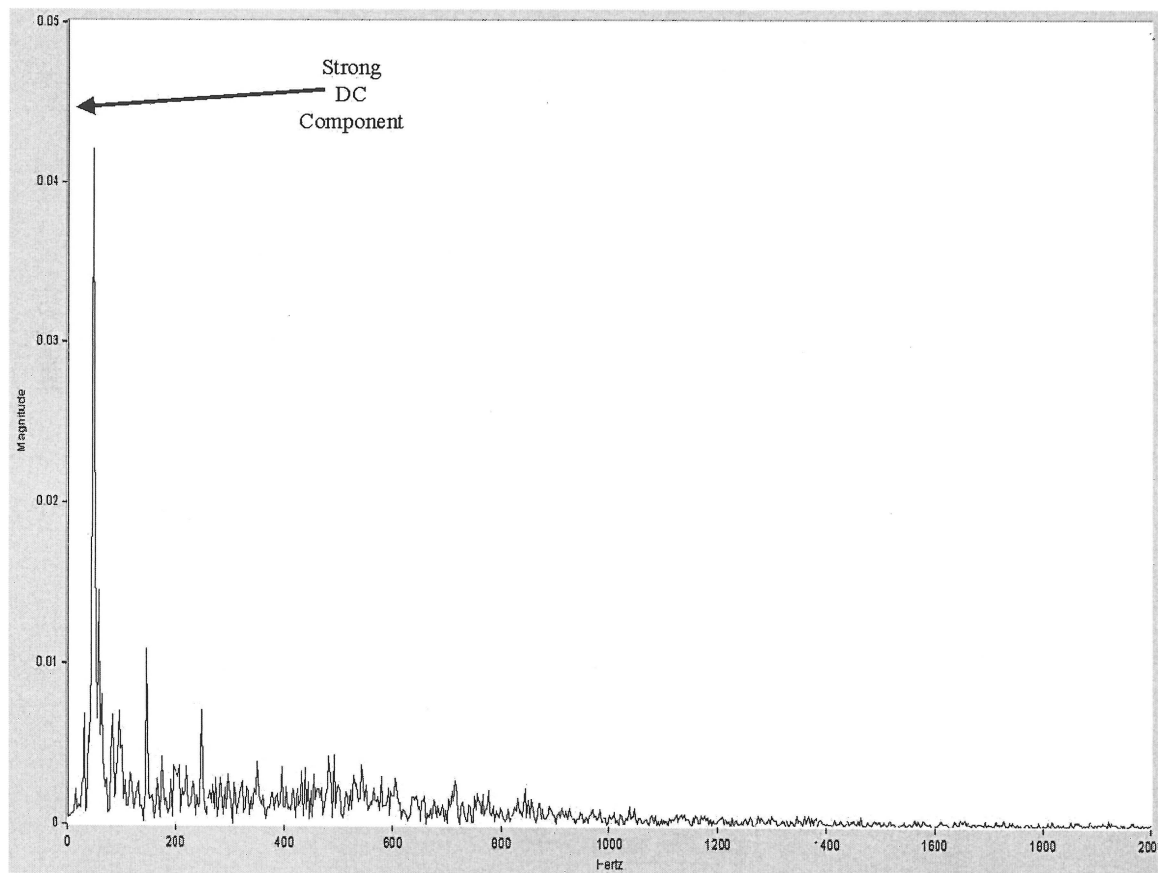
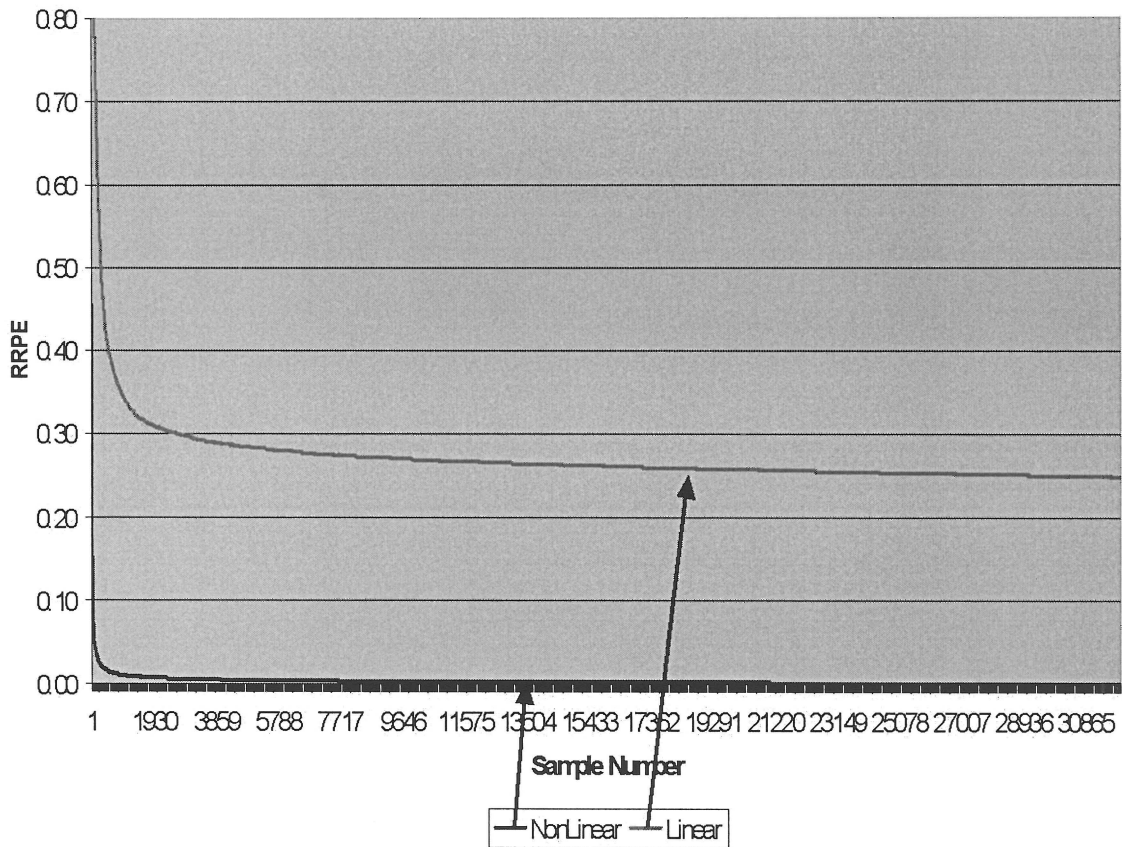


Figure 16. Spectral plot of the chaotic data

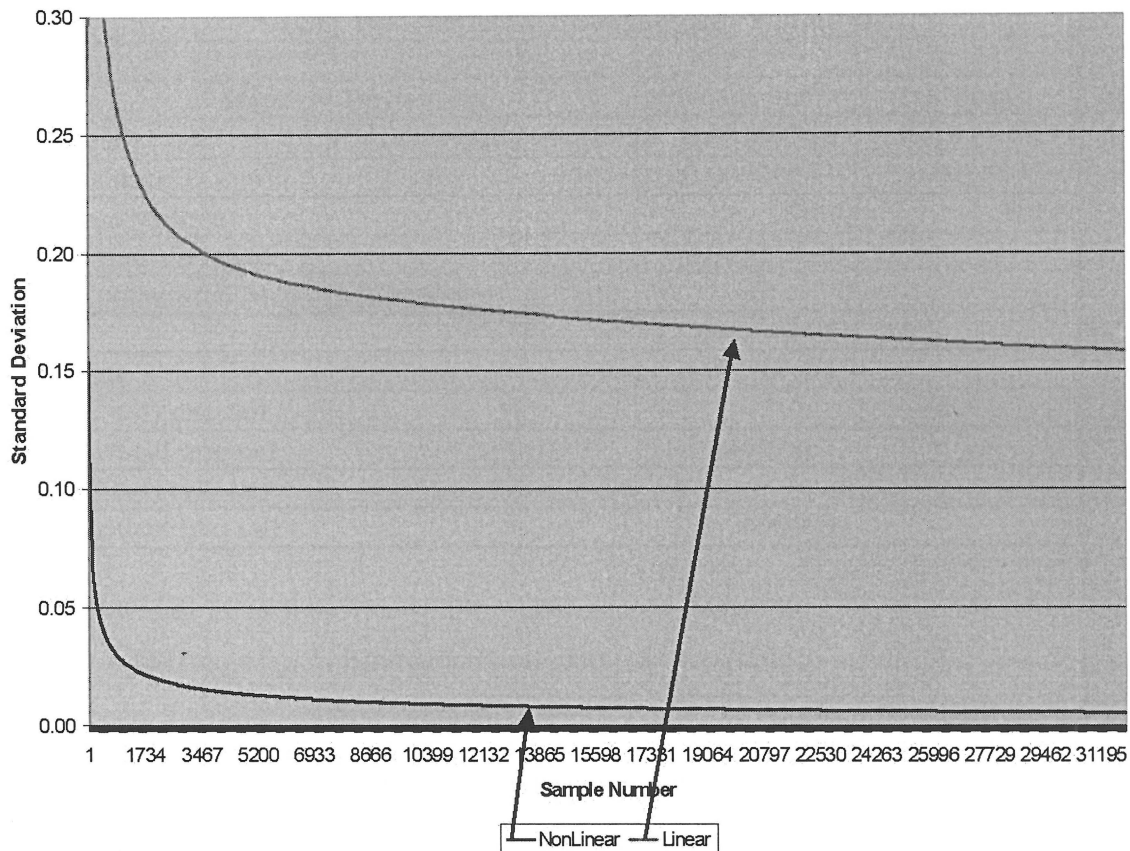


**Figure 17. Spectral plot of the fan noise data**

The chaotic data was already in the range 0 .. 1 so it was not necessary to perform an experiment with mapped data. A comparison of the range-relative prediction errors averaged over time is shown in Figure 18. Figure 19 contains the evolving standard deviation.



**Figure 18. Comparison of range-relative prediction error for chaotic data**



**Figure 19. Comparison of standard deviation for chaotic data experiment**

Statistical tests showed that there is a significant difference for both range-relative prediction errors and standard deviations. The following two tables offer a summary of these calculations.

t-Test: Two-Sample Assuming Equal Variances		
RRPE	NonLinear	Linear
Mean	0.018783761	0.161917152
Variance	0.000368539	0.009415051
Observations	32000	31999
Pooled Variance	0.004891724	
Hypothesized Mean Difference	0	
df	63997	
t Stat	-258.8609747	
P(T<=t) one-tail	0	
t Critical one-tail	1.644878012	
P(T<=t) two-tail	0	
t Critical two-tail	1.96000201	

t-Test: Two-Sample Assuming Equal Variances		
Standard Deviation	NonLinear	Linear
Mean	0.031075439	0.141569691
Variance	0.000766539	0.001463286
Observations	32000	31999
Pooled Variance	0.001114907	
Hypothesized Mean Difference	0	
df	63997	
t Stat	-418.5784545	
P(T<=t) one-tail	0	
t Critical one-tail	1.644878012	
P(T<=t) two-tail	0	
t Critical two-tail	1.96000201	

The original fan-noise data has sample values all less than zero. This data was first mapped to the range 0 .. 1. Both prediction methods were able to handle that data. Figure 20 compares the range-relative prediction errors. Figure 21 compares standard deviations.

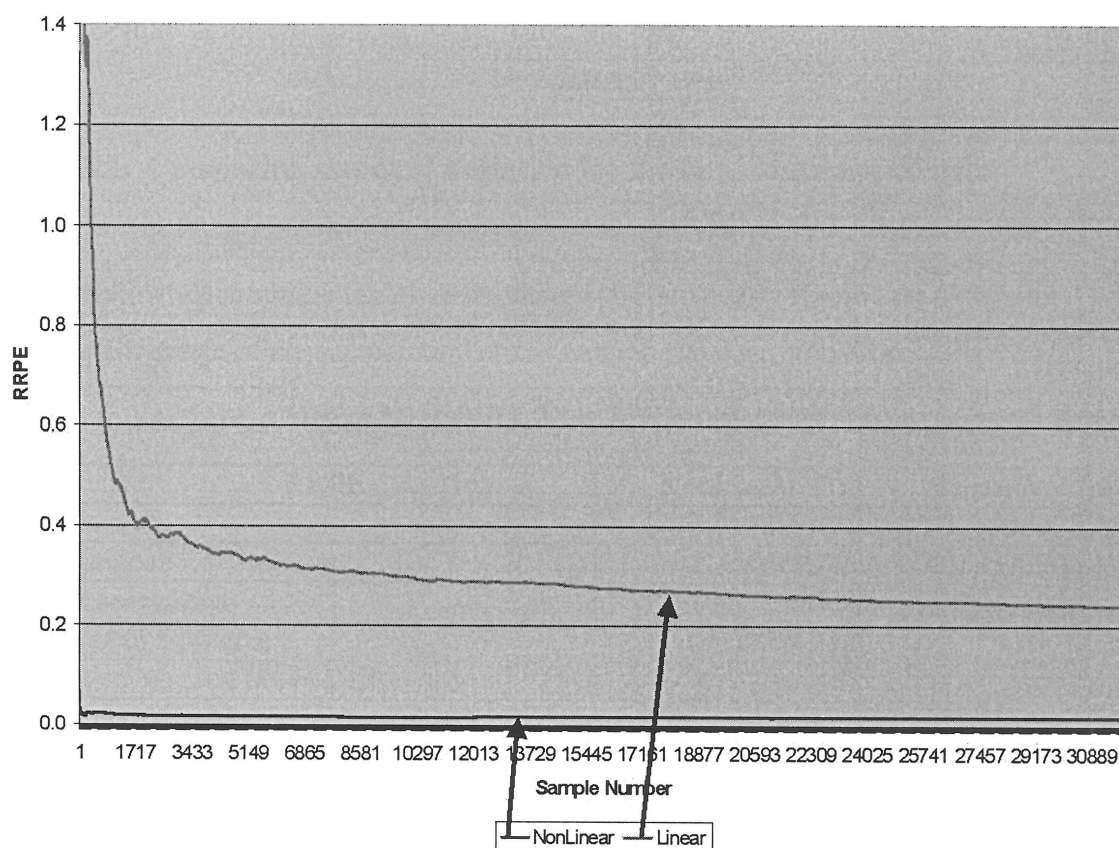
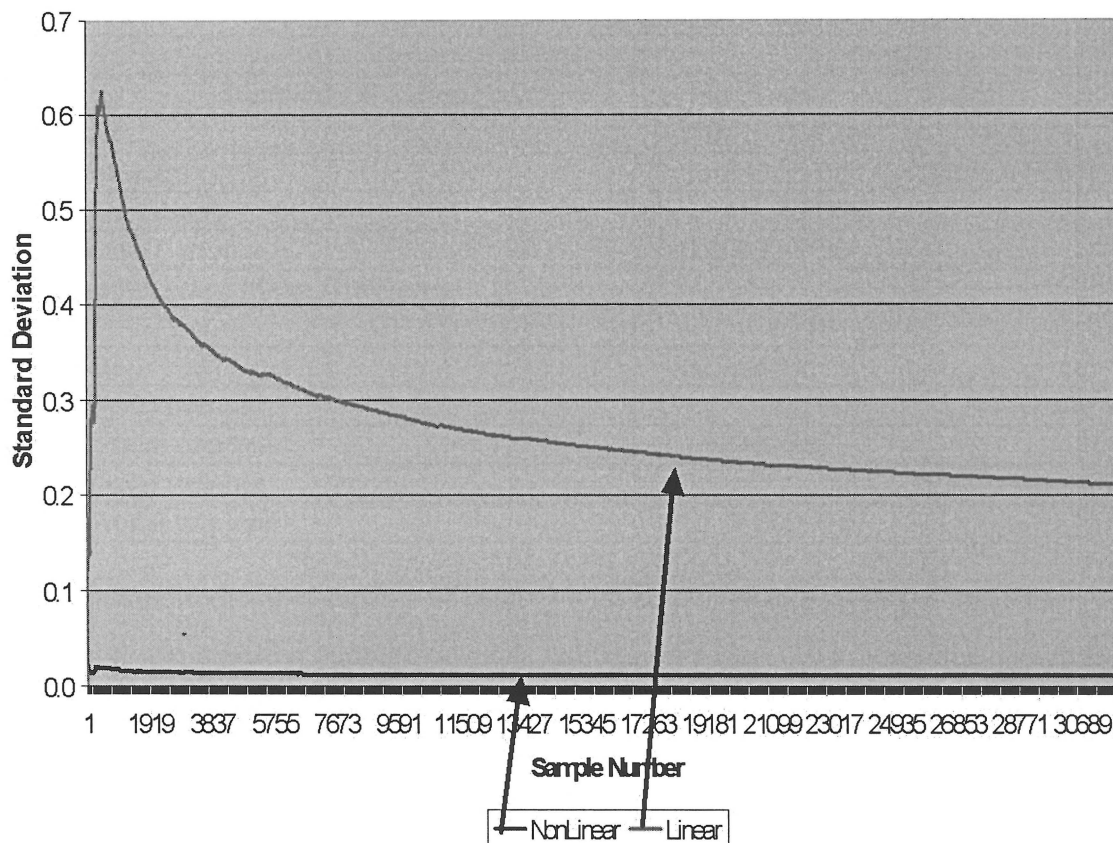


Figure 20. Comparing range-relative prediction error for fan-noise mapped data



**Figure 21. Comparing standard deviation for the fan-noise mapped data**

As the following summary tables show, there is a statistically significant difference between the range-relative prediction errors and the standard deviations.

t-Test: Two-Sample Assuming Equal Variances		
RRPE	NonLinear	Linear
Mean	0.015887637	0.307905233
Variance	1.82066E-06	0.018572305
Observations	32000	31999
Pooled Variance	0.009286918	
Hypothesized Mean Difference	0	
df	63997	
t Stat	-383.2920573	
P(T<=t) one-tail	0	
t Critical one-tail	1.644878012	
P(T<=t) two-tail	0	
t Critical two-tail	1.96000201	



t-Test: Two-Sample Assuming Equal Variances		
Standard Deviation	NonLinear	Linear
Mean	0.012918234	0.274495128
Variance	1.43875E-06	0.005638906
Observations	32000	31999
Pooled Variance	0.002820128	
Hypothesized Mean Difference	0	
df	63997	
t Stat	-623.0480578	
P(T<=t) one-tail	0	
t Critical one-tail	1.644878012	
P(T<=t) two-tail	0	
t Critical two-tail	1.96000201	

The Gaussian predictor is capable of dealing with any range of data so the fan-noise experiment was repeated with the raw data unmapped. Figure 22 contains the comparison of range-relative prediction error while Figure 23 compares the standard deviation.

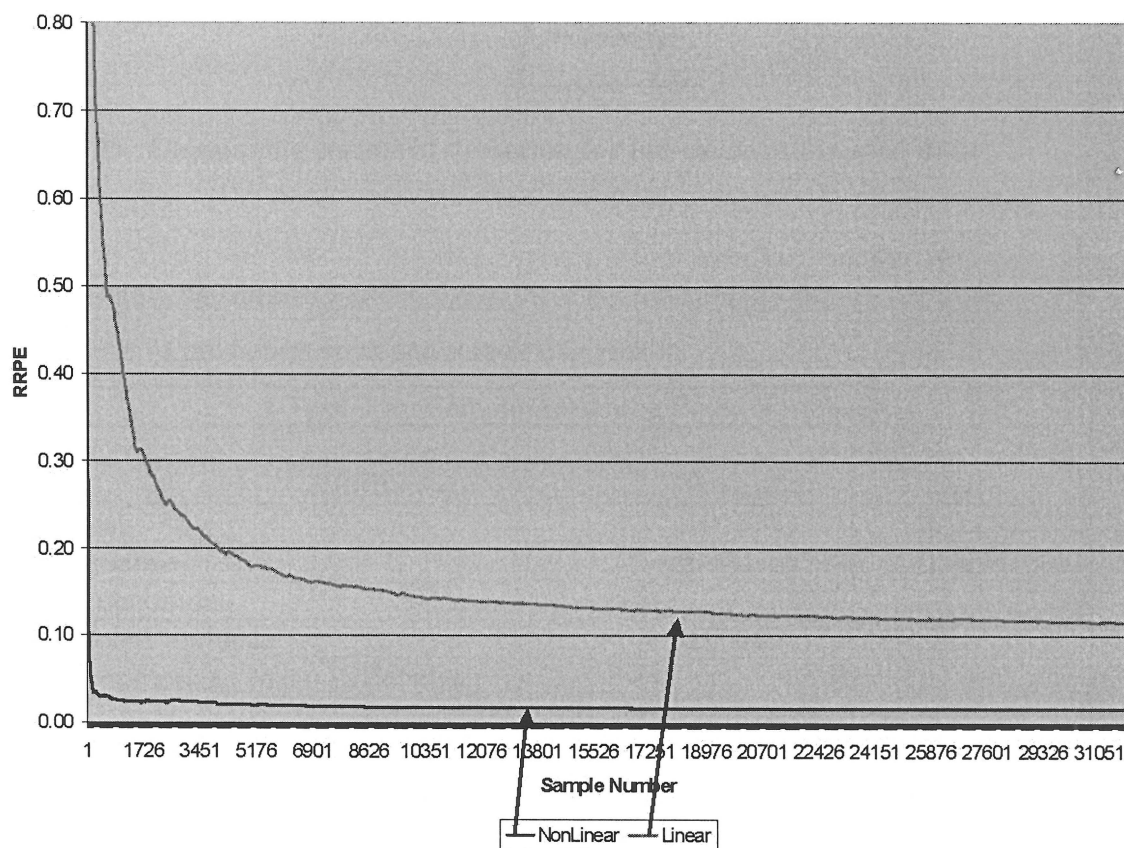
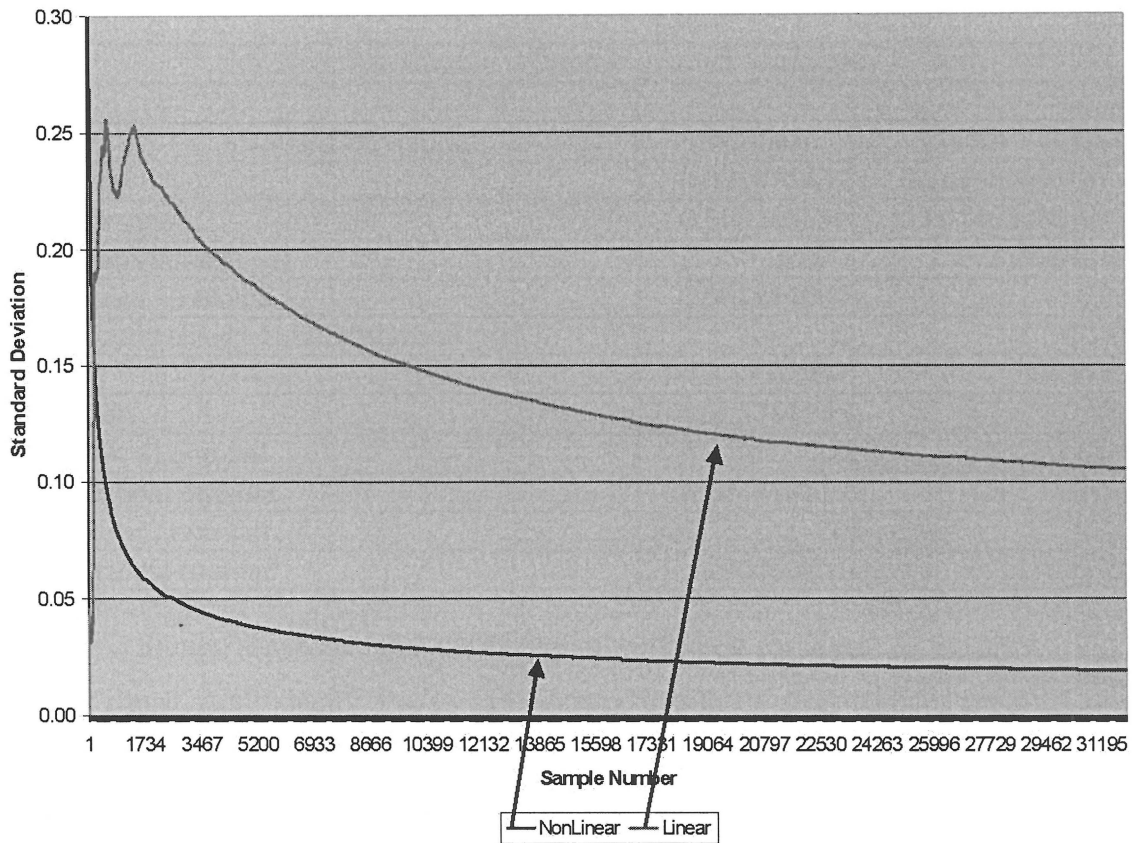


Figure 22. Comparing range-relative prediction error for fan-noise unmapped data





**Figure 23. Comparing standard deviation for fan-noise unmapped data**

As the following summary tables show, there is a statistically significant difference in range-relative prediction error and standard deviation.

t-Test: Two-Sample Assuming Equal Variances		
RRPE	NonLinear	Linear
Mean	0.018783761	0.161917152
Variance	0.000368539	0.009415051
Observations	32000	31999
Pooled Variance	0.004891724	
Hypothesized Mean Difference	0	
df	63997	
t Stat	-258.8609747	
P(T<=t) one-tail	0	
t Critical one-tail	1.644878012	
P(T<=t) two-tail	0	
t Critical two-tail	1.96000201	

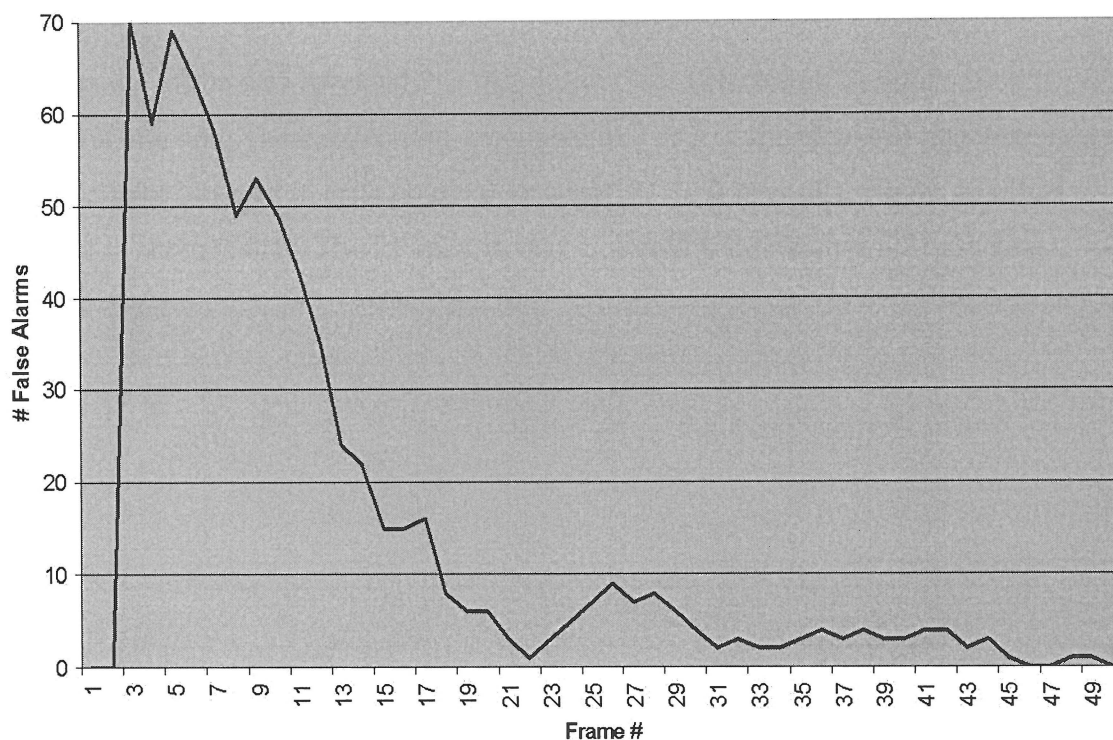


Figure 40. False Alarms prior to intruder entering the chaotic scene

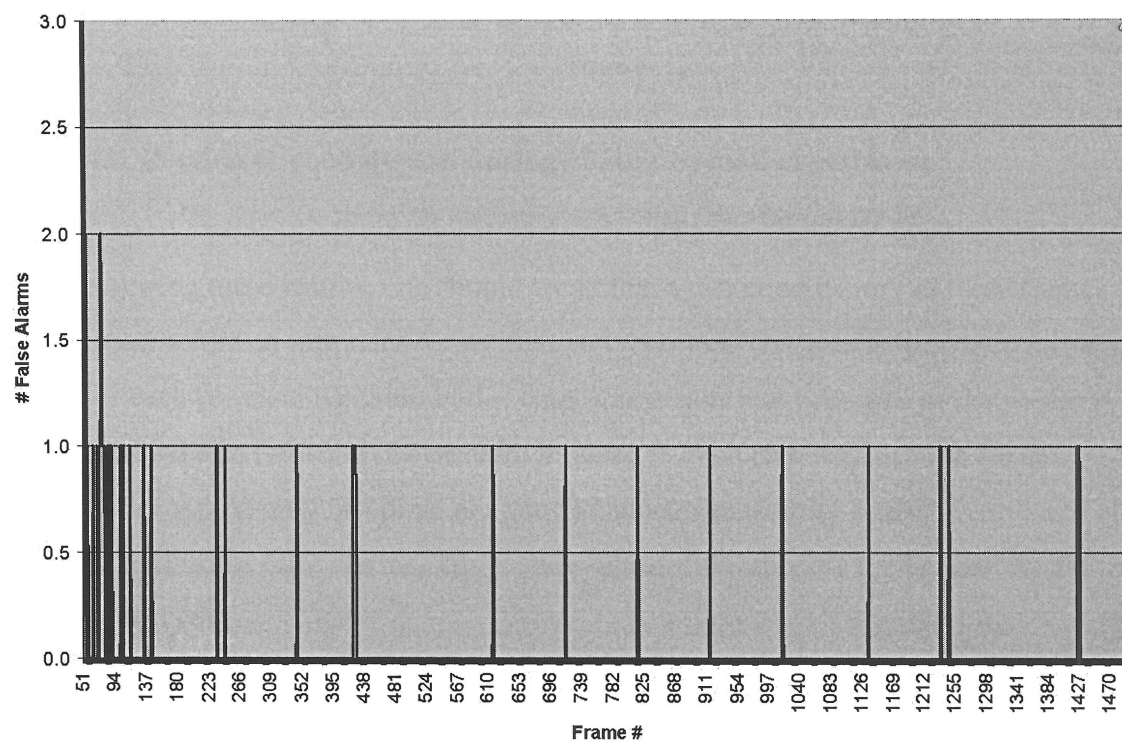
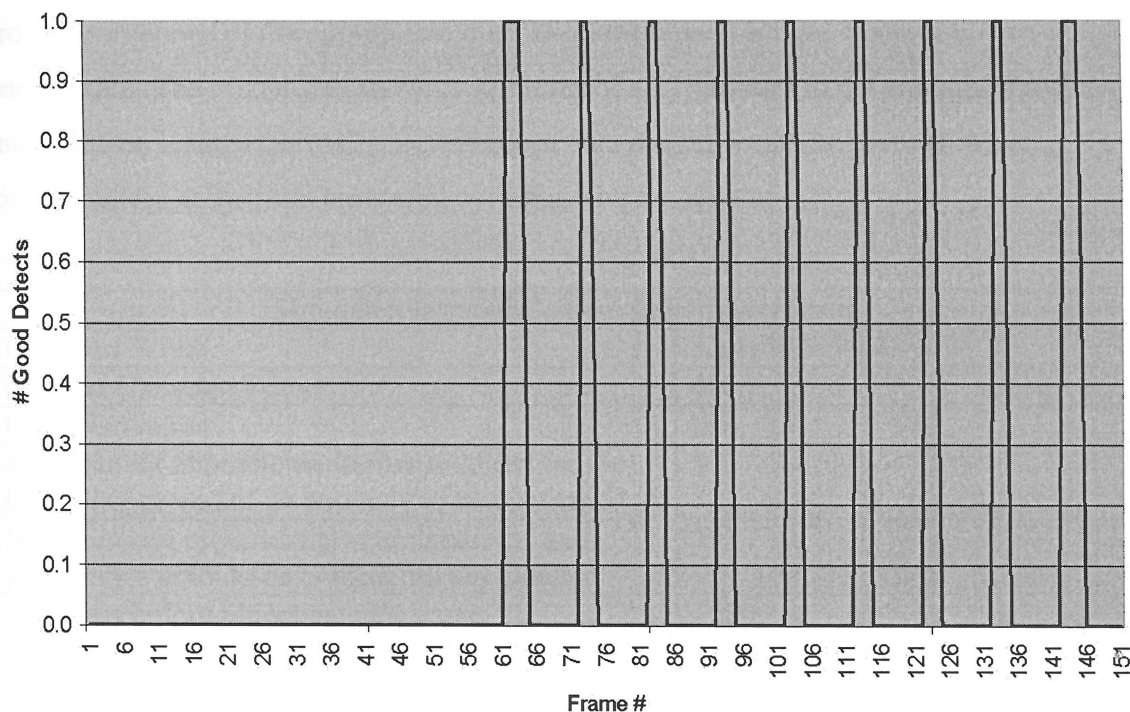


Figure 41. False alarms after intruder entered the chaotic scene

Examination of the data revealed that the intruder was detected in all but the first physical location it attained. These detections amounted to 2 or 3 of the 10 pixels that were taken as the intruder lingered in each physical location (21.31% over all). Figure 42 offers a snapshot of these results. The full plot is very cluttered and therefore not revealing.



**Figure 42. Portion of good detects during chaotic eyeball experiment**

When reviewing these results, one should recall that a detection history of three frames was required before an intruder was verified and reported. This guarantees three missed detects for each physical location. In this case, this means that one-third of the possible good detects will be missed. (The intruder lingered in each physical location for ten frames and occupied only one pixel at a time.) Detector sensitivity could be increased at the risk of additional false alarms. The requirements of the specific application would drive this decision.

For the first thrust at real-world data, a demonstration using frame data was performed by generating pixels from a proprietary infrared camera simulator. The simulator also

produced a very small amount of platform vibration (jitter). The goal was to automatically build an adaptive model that filters out the background and leaves only the intruder on the display. Given available equipment, only a 10x10 frame was simulated.

Each pixel had its own detection model. Each of the 100 models were built automatically from observations of the appropriate pixel in 75 sequential frames of intruder-free background. The model was shown 100 cycles of all 75 frames to get enough exposure to intruder-free scenes. Normally, a continuous data stream would be used. However, circumstances at the time prevented our obtaining more data.

Simulated Infrared Camera Experiment Setup	
Gaussian Width	3.141592654e4
Number Gaussian Terms	800
Gain Numerator	0.5
# Sequential unpredicted frames required for the beginning or end of an anomaly to be confirmed	4
% Difference required between predicted and actual for a pixel to be considered unpredicted	20
# Frames before object entered scene	7500
# Frames before model entered detection mode	7500
Moving average length	1

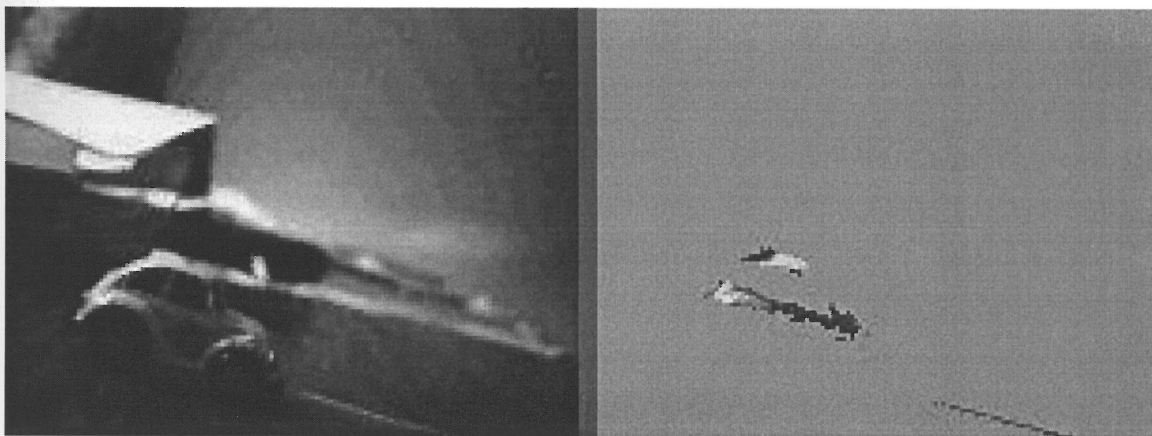
Out of the series of frames that the object lingered at any one pixel, 4 frames were missed because there had to be 4 in order to confirm the presence of the anomaly. For each of ten frames, there were 4 false alarms since 4 were needed to confirm the end of the anomaly. Another false alarm (relative to the actual intruder) was a two-pixel hot-spot developed by the simulator in another part of the frame from the intruder's track. The intruder's first pixel position was missed altogether. A precise percentage of false alarms and missed detects was not possible to calculate due to the nature of the data.

Two results from QuickCam scenes are discussed next. The following table gives the configuration of the detector for both the benign and dynamic cases.

Two-Dimensional Data Stream Experiment Setup	
Gaussian Width	0.0011286596702183
Number Gaussian Terms	30
Gain Numerator	0.1
# Sequential unpredicted frames required for the start / end of an anomaly to be confirmed	3 / 1
% Difference required between predicted and actual for a pixel to be considered unpredicted	9.5
# Frames before object entered scene	30 data minutes
# Frames before model entered detection mode	30 data minutes
Moving average length	1

During the experiment with a benign scene, a model car was pulled through the scene. The detector, according to a visual estimate, was able to turn on an average of 50% of the pixels associated with the car as it moved through the scene. Visually, there were no false alarms and no lighting changes were detected. The model did detect the string as it changed position while pulling the car.

The model learned to predict the base scene within five data seconds. Figure 43 compares<sup>®</sup> one frame of the unfiltered and filtered scene as the car was moving through the scene.



**Figure 43. Unfiltered and filtered benign scene as model car moves in the scene**

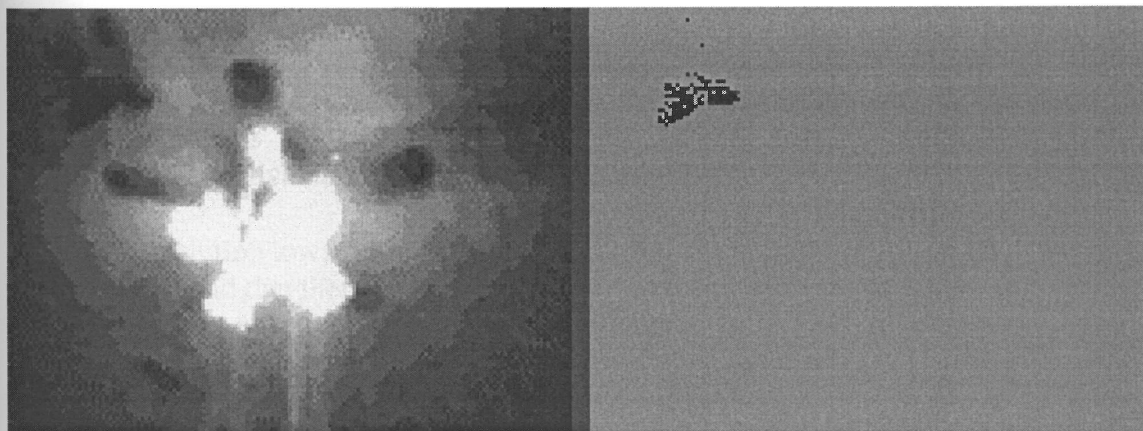


While there were no visually detectable false alarms, detailed examination of the output data stream showed that there were some single-pixel false alarms. These were so small and brief in the scene that the eye did not catch them. In this sense, the eye essentially acted as a spatial filter and ignored single-pixel false alarms. An automatic filter could also be developed to do that as a post-processor for the detector. A choice to do something like that would depend on the application.

Empirical observation showed that more pixels could be turned on by the model at the risk of increasing the number of false alarms. The only change in the model to accomplish this would be to decrease the threshold between the average and instantaneous RRPE. Another way to turn on more pixels would be to use a clustering method such that if some number of pixels were turned on within a cluster boundary then all the pixels within that boundary would be turned on. As shown in the detection experiments with single-dimensional data streams, there are many ways to configure the detector depending on the needs of the application.

The final detection experiment was with the dynamic QuickCam scene. A model airplane was pulled below a spinning ceiling fan. The scene was undersampled so that the fan appeared to be turning in random directions. A light was attached to the fan below the blades. The light was turned on.

The model learned to predict the scene within 30 data minutes. Visually, the model turned on an average of 50% of the pixels associated with the airplane as it was pulled through the scene. Visually, there were no false alarms. Numerous lighting changes were detected. These changes occurred because the airplane crossed between the fan's light and the camera. The model did not detect the string as it changed position in the process of pulling the airplane. Figure 44 compares one frame of the unfiltered and filtered scene as the airplane was being pulled through the scene.



**Figure 44. Unfiltered and Filtered robust scene as model plane moves in the scene**

The same discussion on false alarms and turning on additional pixels applies here as with the benign QuickCam scene. It should also be noted that, in the case of the dynamic QuickCam scene, the model detected evidence of the anomaly's presence along with the anomaly itself. The lighting changes were caused by the presence of the anomaly.

### **Results from the Implementation Experiments**

As demonstrated by the results from the experiments with the theory, PAD works well with a variety of synthetic and natural data. The main point of the research is concluded thereby but the author wanted to take things a bit further and demonstrate the potential for implementation in practical settings. Thus, it seemed appropriate to explore alternatives for improving throughput.

Recognized immediately is that PAD is CPU intensive. (Yet, it scales linearly.) The implementation experiments took advantage of PAD's naturally-parallel architecture while exploring three opportunities to improve throughput:

- code improvements on a single-cpu/single-machine PC
  - these had a positive affect but were not statistically significant

- parallel processing on a multi-cpu/single-machine and a multi-machine cluster
  - these had a positive affect but were statistically significant only for large problems
  - takes advantage of course-grain parallelism
- evolution toward FPGA implementation
  - found that there is a huge potential but that a software paradigm shift needed
  - takes advantage of fine-grain parallelism

The parallel processing experiments split PAD by numbers of pixels. Each CPU's part was to for evaluate and update the model terms for a given number of pixels. Overhead was evaluated as the number of CPUs and machines expanded. Left to future work was pipelining for such things as data acquisition, distribution, display, and storage.

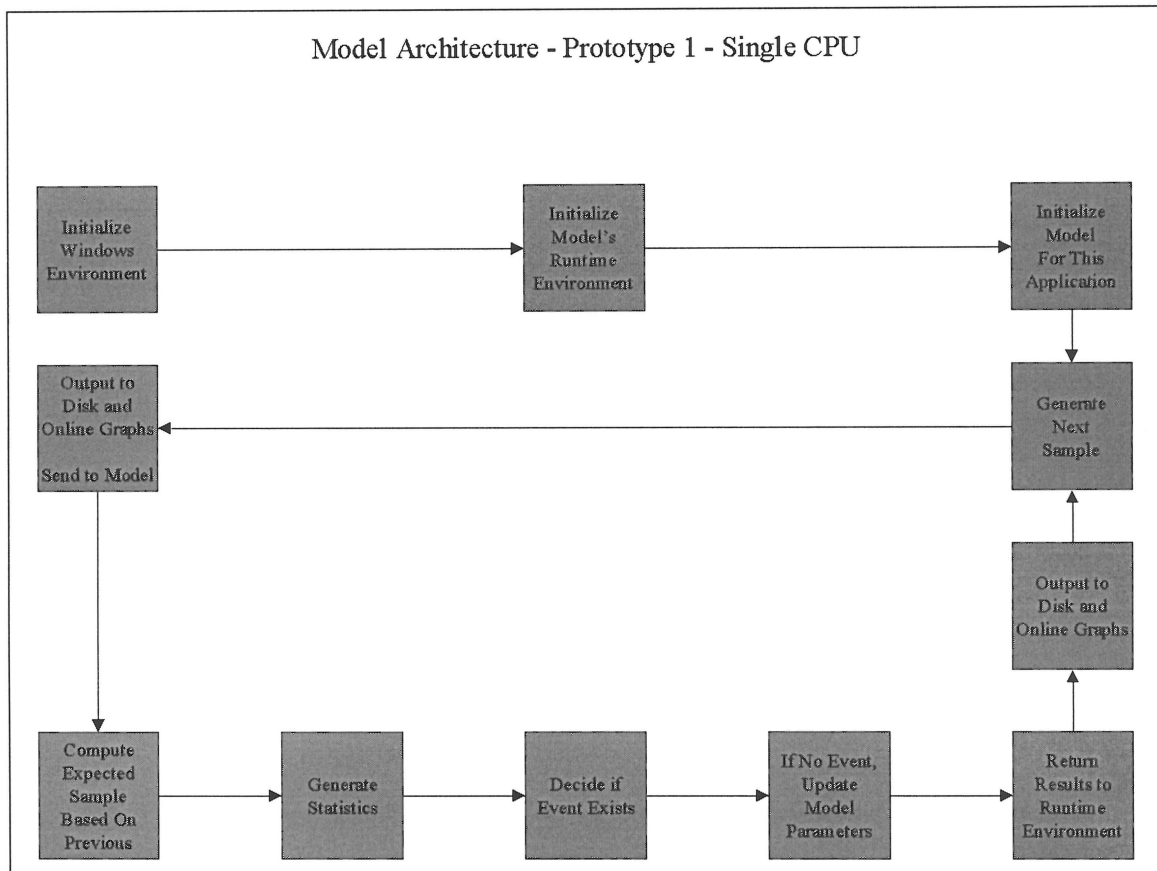
### Single-Machine Single-CPU Throughput Experiment

#### *The Original Trial*

After the code was instrumented to allow the collection of metrics, the original trial with the simulated IR-camera was run again so that specific data could be collected on throughput and a baseline established. All comparisons were made to the baseline. Nothing else was running on the computer. This trial took 2765 seconds to complete.

The architecture of the model for these single-CPU / single-machine comparison trials is shown in Figure 45.





**Figure 45. Model architecture: single-CPU / single-machine**

### *Comparison Trial 1*

To prepare for the first comparison trial, a copy was made of the instrumented code. Then, a lot of code cleanup was performed:

- A lookup table replaced the constant recalculation of basis function centers. The centers do not change so there is no need to keep recomputing them.
- The function that determines the adaptation signal was modified to permit multiple return points rather than setting a control variable. This function is deep in the loop that iterates through all basis functions. For 100 pixels and 800 hidden nodes in the model for each pixel there are  $100 \times 800 = 80,000$  equations for each frame. Thus, it was felt that improving this function would be worthwhile.

- Compilation and linking used VC++ standard release mode. This removes all the debugging code.
- The code was executed directly from the operating system file menu, rather than under the control of the development environment. This removes a bit more overhead.
- Each frame was read using one statement that inputs all the frame's pixels at once, rather than using a loop to read each pixel one at a time. This saves a loop and multiple file accesses per frame.
- Liposuction  
(<http://www.microsoft.com/msj/defaulttop.asp?page=/msj/archive/s572.htm>) removed excess statements from the executable code.

As a result, the run took 2484 seconds. This is indeed numerically smaller than the original. However, there is no statistical difference in the run times. This was determined by comparing the amount of time to complete each cycle during the original and trial\_1 runs.

There are other code-level modifications that can be tried during future work:

- The exponential is employed heavily. It may be possible to replace the standard call to  $\exp(x)$  with a shortened in-line Taylor series without losing the model's predictive capability. Something like this may tend to run faster. (Suggestion from Dr Steve Gustafson, University of Dayton Research Institute.)
- Opportunities to replace division with multiplication could be explored. There may also be opportunities to perform integer, rather than floating point, arithmetic. Further, one could explore opportunities to use left and right bit shifts instead of multiplication and division by powers of 2. (Suggestions from Charlie Todd, Ball

Aerospace & Technologies Corp.)

- Higher-grade compilers have options that could improve the optimization of the object code. (The student version of VC++ has only a default optimization whereas the professional version has several additional capabilities. However, it is necessary to reckon with the testing needed to verify that the optimized code performs the same way functionally as the non-optimized code.)
- Some of the code be replaced by assembler routines and other machine-specific improvements. A profiler would show where this and other attempts at speeding up the code might be most profitably applied. However, these types of changes interfere with the generic nature of the code and relegate it to a specific hardware suite.

### *Comparison Trial 2*

Comparison trials 2 and 3 set aside attempts at improving the code and looked at the basic algorithm used to construct and evolve the model.

Since the data being used in these trials is very robust, one might assume that a large model is needed. Of course, the larger the model, the longer the run time and the lower the throughput. This trial took the original experiment and decreased the number of nodes until just before a statistically significant increase in prediction error occurred. The model reached this point at 775 nodes. This resulted in a 26% decrease in run time. Comparison showed that this is statistically significant.

### *Comparison Trial 3*

Setting the gain numerator to 0.05 in the trial\_2 code, it was possible to bring the number of nodes down to 25 before a statistically significant decrease in prediction accuracy

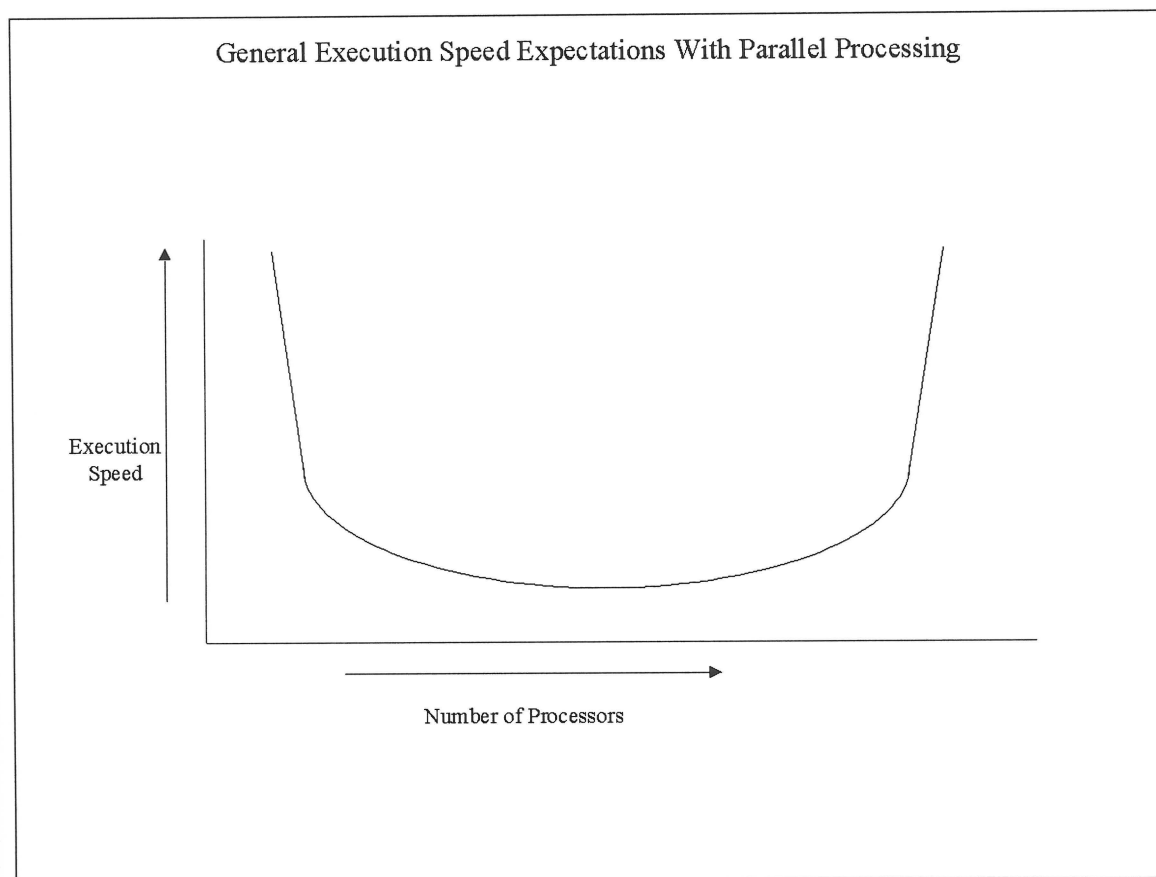
occurred relative to the original trial. The run took 77 seconds, a statistically significant improvement in throughput.

### **Multi-CPU Throughput Experiments**

The next thrust in throughput improvement made use of a multiple-CPU architecture. This is not a cure-all as the following discussion shows.

#### *Issues in Parallel Processing*

By having several CPUs perform smaller parts of a single large problem, the hope is that the problem gets solved faster. That this is typically true is demonstrated by Sevenich (1998) and by Souza, Senger, Santana, & Santana (1997). It should be noted, however, that for any given problem and architecture there is a limit to how much improvement can be had by simply adding more processors. This is because of contention for the memory bus (in a shared-memory environment), contention for the network (in a distributed-memory environment), the need for processor synchronization, and portions of the code that must be run sequentially. A typical performance curve is shown in Figure 46. Morse (1994, p 32-43), Toxvaerd (1994, p 503), Mattson (1996, p 992), and Stewart (1994, p 244) show applications that exhibit this curve's general shape. Figure 46 shows that, for a given problem and architecture, adding more processors helps a great deal at first. Then the benefit from additional processors ceases to increase. With still more processors, resource contention slows the application down until the relationship between the number of processors and the throughput is increasingly inverse.



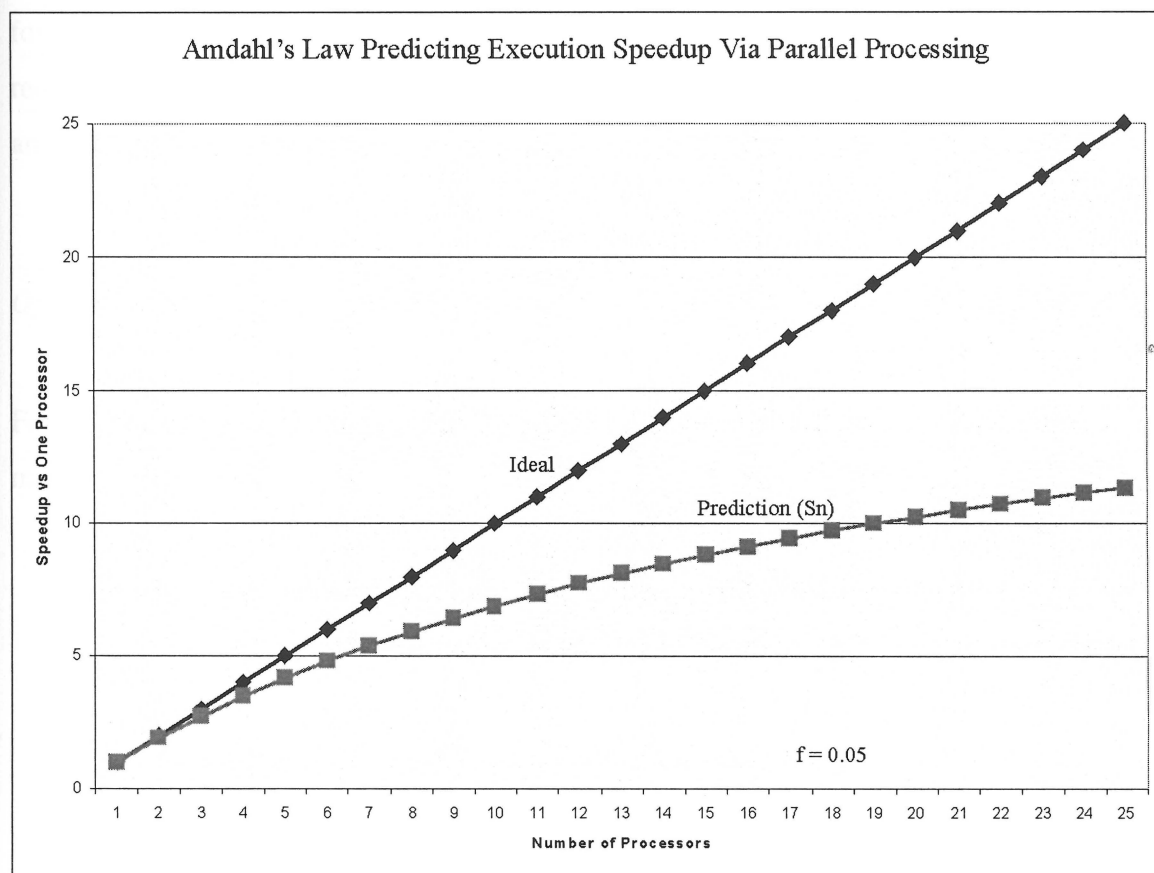
**Figure 46. General Execution Speed Expectations With Parallel Processing**

Amdahl (1967) derived an equation that predicts the amount of speedup to be expected from a number of processors given a fraction of code that has to be run sequentially (cited and explained by Zomaya, 1996, p 14; see Appendix C for more details).

$$S_N \leq \frac{1}{(f + ((1 - f) / N))}$$

Where:  $S_N$     the predicted execution time divisor for the single-processor case  
            $N$         the number of processors  
            $f$         fraction of the code that must be executed sequentially  
                       (loops unwound)

This equation is illustrated in Figure 47 for the case  $f = 0.05$ . Note that Amdahl's equation only deals with the fraction of code that must be executed sequentially. It does not take into account the other drawbacks noted earlier. However, his equation does tend to confirm the curve in Figure 46. Memory bus and network contention, along with processor synchronization, are the overhead that makes execution time begin to slow down again once too many processors are included. Note also the small fraction of sequential code, Figure 47 assumes ( $f = 0.05$ ). The message here is that a program has to be carefully examined and arranged so that it takes best advantage of multiple CPUs. The "code fraction" is not the popular "executable lines of code" used in cost estimating. If 90% of the executable lines of code must be run sequentially but only 5% of the time is spent there then  $f = 0.05$ .



**Figure 47 Amdahl's Law Predicting Execution Speedup Via Parallel Processing**

Another issue is problem size. For problems that are too small, parallel processing can decrease their throughput. This is due to the fact that the overhead attendant on parallel processing can cause the total execution time to exceed what it would take a single-CPU to complete the run. Going back to Amdahl, if the time spent executing the 10% of the code that can run in parallel is smaller than the time spent in the 90% that must be run sequentially, the run-time due to the sequential code and the parallel-processing overhead will dominate the throughput and slow things down. Pipelining is one option that can alleviate this problem. This technique uses an architecture of several processors but each processor performs a larger portion of the problem as part of a sequence. For instance, next-sample input and preprocessing can be performed at the same time as current-sample final processing and output. This does, however, introduce additional overhead for processor synchronization. Thus, the use of parallel and distributed processing requires a constant balance of application, architecture, CPU demands, memory demands, and data-passing capacity.

### *Opportunities*

Four opportunities were considered for parallel operation of the prediction/detection model:

- Split the model's main loop. The number of parameters in the model determines the length of this loop. The loop can be split because each model parameter is independent of all the others. Currently, the number of parameters is fixed prior to the start of processing. Even so, variable loop sizes can be accommodated. The code for each loop index can execute independently on its own processor.

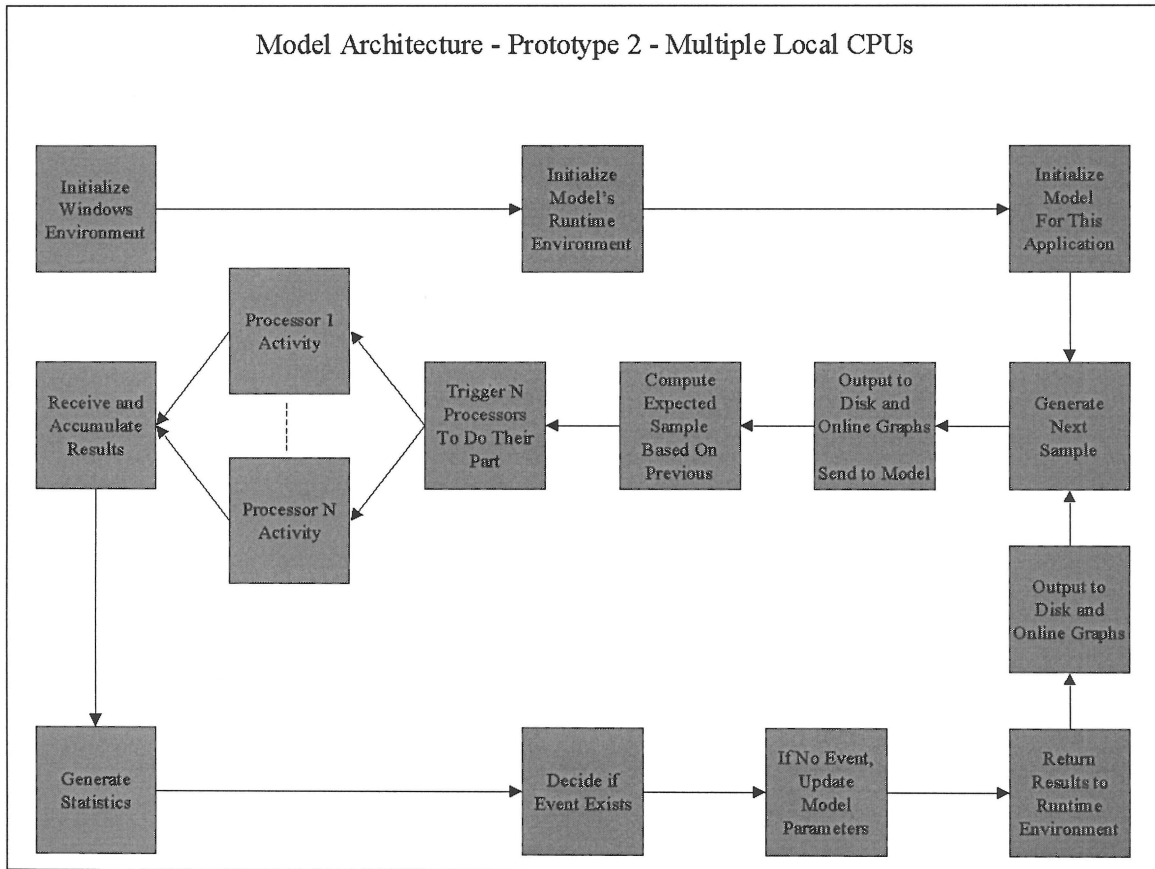
There are many approaches to splitting loops. One approach is offered by Letner (1996). He has each processor handle a given pass through all loop indexes. This works well when the code for each loop index is dependent on

the other indexes but each (entire) loop produces independent results. An approach used by Pilati (2000) has each processor execute a given portion of the loop. This is effective when each loops' results are dependent on the results from previous passes through the loop but the code for each loop index is independent of the code for other indexes.

- Generate the model's output as a separate process.
- Update the model using another separate process.
- Implement model input, output, and update as steps in a pipeline.

When moving in the direction of parallel / distributed processing it is important to provided additional efficiencies without undoing the flexibility of the implementation. The basic idea used here was to first use Pilati's (2000) method to split the model's main loop. This architecture is illustrated in Figure 48 and takes advantage of multiple CPUs on the same machine. It is an evolution of the architecture used for the original single-machine/single-CPU comparison tests (see Figure 45).

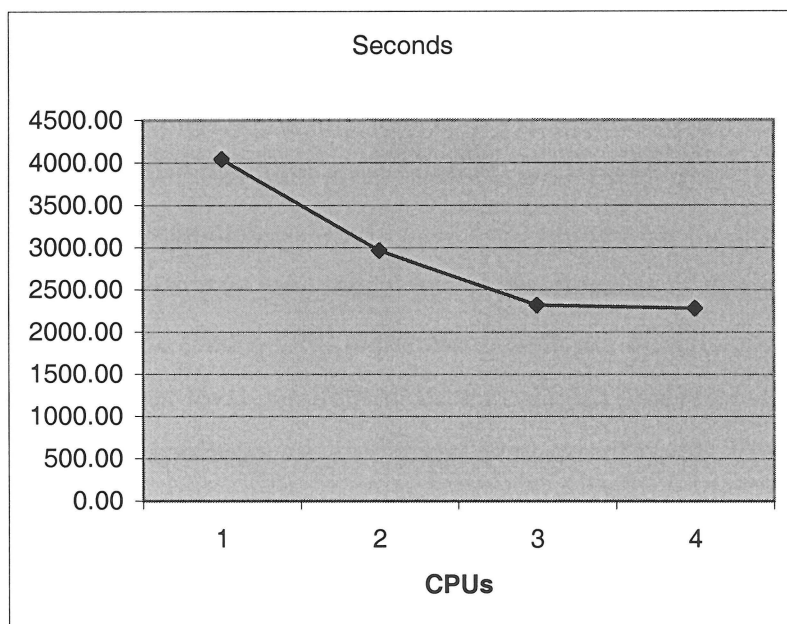




**Figure 48. Model architecture: single-machine / multiple-CPU**

### *Experiment with a Uni-Dimensional Data Stream*

A uni-dimensional data stream was applied to a model with 25,000 terms. The single-machine / multiple-CPU architecture of Figure 48 was used. Preliminary experiments with lesser numbers of terms showed throughput improvements that numerically improved as the number of CPUs increased from 1 though 4 but no statistically significant improvement was observed. The results from the 25,000-term experiment are summarized in Figure 49.



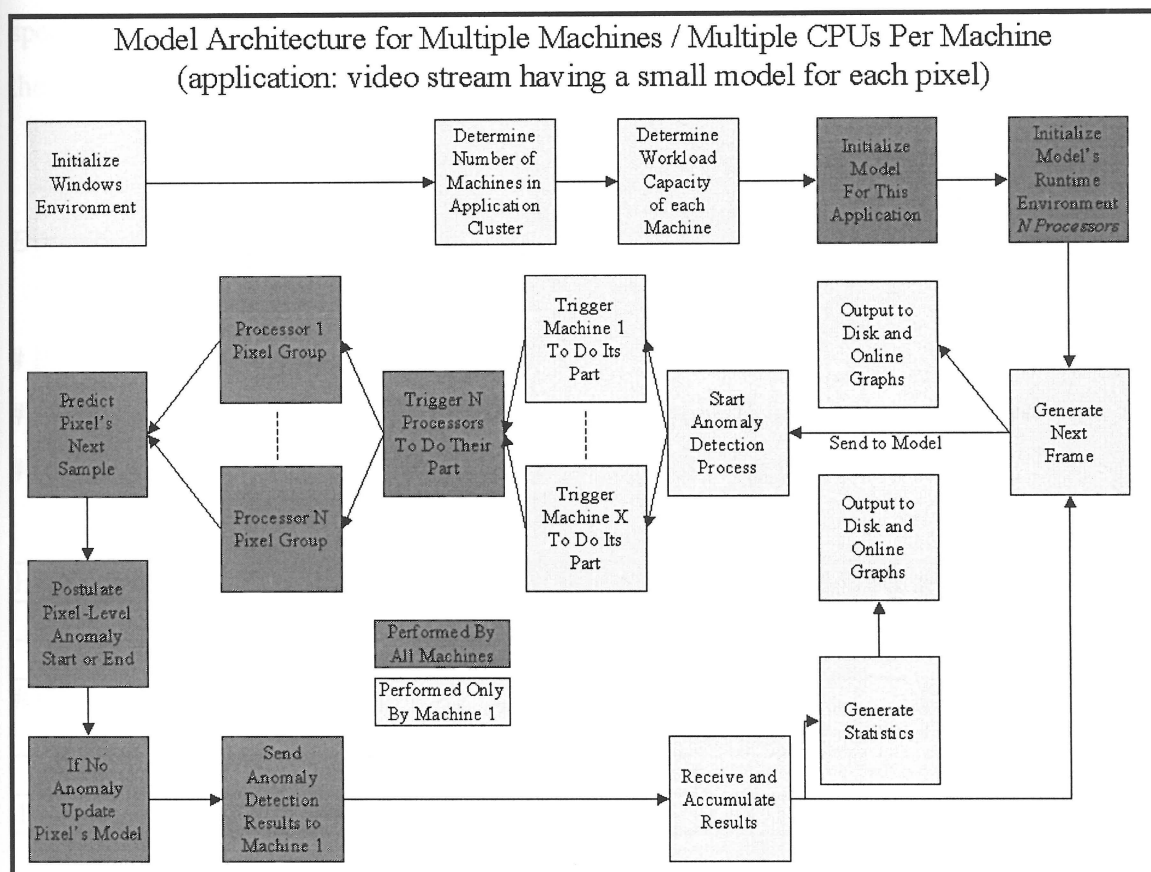
With 25000 terms in the model				
# CPUs	Seconds	Actual Speed-Up	Efficiency	Estimated f
1	4042.69			
2	2966.23	1.36	0.68	0.47
3	2319.28	1.74	0.58	0.36
4	2282.83	1.77	0.44	0.42

**Figure 49. Results from experiment with loop-splitting in a single-machine / multiple-CPU architecture**

### Multi-Machine / Multi-CPU Experiments

The effort in throughput improvement took its next step by applying the model to the more complex image stream problem. A total of 8 processors were employed by adding an additional 4-processor computer to the mix. This created a parallel/distributed cluster where each group of four processors has its own shared memory. This approach enabled reasonable throughput improvement for the much-larger image-stream models. It also offered a chance to examine the issues brought about by inter-computer communication.

A major lesson learned from the uni-dimensional data-stream experiment reported above is that the model has to have at least 25,000 terms for parallel processing to produce statistically significant throughput improvement. The earlier experiments with image stream prediction showed that each pixel needed a model with only 20 terms. Thus, loop-splitting for term-level parallelism would have been ruinous for the many small models used to process image streams (one model per pixel). Because of this, the architecture used for single-machine / multi-CPU experiments with uni-dimensional data streams was modified to enable pixel-level parallelism. Instead of each processor receiving a certain number of terms, the processors are each given a number of pixels. The terms for a given pixel are all computed and updated by a single processor. If a processor is located on another computer, the appropriate part of the image is sent to that computer. From there the image chip pixels are distributed to the various processors. The results are returned to the host computer for accumulation and posting. The resulting multi-machine/multi-CPU architecture is illustrated in Figure 50. (Recall that the number of processors on each computer need not be the same. The software is set up to automatically adjust to the number of computers and processors, and to the workload capacity of each computer.)



**Figure 50. Architecture for applying the model to image stream pixels**

This architecture implies the need for a message-passing infrastructure. While one can easily argue that a custom-written infrastructure would serve the needs of throughput most effectively, the author chose to use a generic infrastructure to preserve the generic nature of the overall detection environment. After considering the various alternatives (Raeth, 14 Nov 2000), the author settled on an implementation of the Message-Passing Interface. MPI (see Numerous Authors, 1998) is an internationally accepted and coordinated industrial standard for passing messages between machines in a heterogeneous cluster. (This means that the machines in a cluster do not have to be all of the same type nor even running the same operating system. They also do not have to be physically local to each other.) Of the academic implementations of MPI, the one that appears most complete is produced by the Center for Scalable Computing at the Aachen University of Technology (see Scholtyssik, 2000). This particular infrastructure is

specifically designed to support machines connected to a common office LAN running the TCP/IP protocol.

### *Experiment 1A*

# Pixels per frame =  $120 \times 160 = 19,200$

# model terms per pixel = 30  $\rightarrow$  576,000 terms to evaluate and update per frame

# frames = 8,290

Experiment 1A: Execution time for the single-machine / single-CPU case					
# CPUs	Seconds	% Due to Overhead			
		A. Frame Receive	C. Frame Store	D. Frame Display	Total
1	16469.2	2.0%	0.7%	22.8%	25.6%

### Experiment 1B

Experiment 1B: Execution time for the single-machine / multi-CPU case							
# CPUs	Seconds	Speed Up	Efficiency	% Due to Overhead			
				A. Frame Receive	C. Frame Store	D. Frame Display	Total
1	16153.7	1.00	1.00	2.0%	0.8%	23.3%	26.0%
2	9366.6	1.72	0.86	3.5%	1.3%	40.1%	44.9%
3	7089.2	2.28	0.76	4.6%	1.7%	53.0%	59.4%
4	5956.6	2.71	0.68	5.5%	2.0%	63.1%	70.7%

### Experiments 1C and 2

Experiment 1C and 2: Execution time for the multi-machine / multi-CPU case								
# CPUs	Seconds	Speed Up	Efficiency	% Due to Overhead				
				A. Frame Receive	B. Inter-Machine Communication	C. Frame Store	D. Frame Display	Total
1	16855.8	1.00	1.00	2.0%		0.7%	22.3%	25.0%
2	9658.4	1.75	0.87	3.5%		1.3%	38.9%	43.7%
3	7260.7	2.32	0.77	4.6%		1.7%	51.8%	58.1%
4	6092.4	2.77	0.69	5.5%		2.0%	61.7%	69.2%
5	5325.7	3.17	0.63	6.3%	3.5%	2.3%	70.6%	82.7%
6	5170.4	3.26	0.54	6.5%	3.6%	2.4%	72.7%	85.2%
7	4809.0	3.51	0.50	7.0%	3.8%	2.5%	78.2%	91.6%
8	4185.5	4.03	0.50	8.0%	4.4%	2.9%	89.8%	105.2%

### Experiment 3

Experiment 3: Determining Overhead						
	A. Frame Read	B. Inter-Machine Communication	C. Frame Store	D. Frame Display	Total Overhead	
Run Time (Seconds)	336.4	185.1	121.8	3760.1	4403.4	
% of Total	7.6%	4.2%	2.8%	85.4%	100.0%	
Avg Time Per Pixel	2.11E-06	1.16E-06	0.0765E-06	20.36E-06	20.77E-06	cluster
Avg Time Per Pixel	2.11E-06		0.0765E-06	20.36E-06	20.65E-06	single machine

## *Discussion*

Note that the fractions of overhead found in the 8-CPU case adds up to more than 100%. This shows that there is unmeasured overhead attributable to the operating system. Still, the numbers are useful for judging relative overhead. There is a certain overhead that will always be present, regardless of the activity performed. This overhead appears in all four phases of experiment three but only once during a full run of the model. The total overhead, however, is estimated by summing the execution times of all four phases of experiment three. Thus, operating system overhead is included four times in the estimate of total overhead. That continuous overhead is present, even when no user programs are running, is verified by the operating system's task manager. It estimates continuous overhead at 1% of CPU resources due to the operating system's 20 tasks. Task overhead from user applications adds to continuous overhead. A real-time operating system may minimize the impact of system overhead on model throughput. A bare-bones version of the general-purpose operating system is another option. In this case, WinNT, the bare-bones version is WinCE. Another option is to provide a co-processor for CPU-intensive parts of the model. This co-processor would be a hardware device executing outside the operating system's management and in parallel with overhead tasks.

It goes without saying that the biggest hold-back to improving throughput at this point is the display of results. The operator will want to see the filtered and unfiltered scene. This means that each frame has to be displayed twice. Since each pixel is assumed to be of a different color, the display is generated pixel by pixel. It would be worthwhile seeing if there are other ways of generating the display. When this work started only one machine and one processor were available. Thus, the percentage overhead devoted to frame display did not appear to be an issue. When viewed from the perspective of a faster cluster, this would be a good place to spend addition design and development time. Leaving such a large block of code in a purely sequential part of the architecture makes a heavy contribution to the loss of efficiency (down to 50% for the 8-CPU case). For the multi-CPU case a thread could be started that would accept pointers to the filtered and



unfiltered frames and then generate the display. This would put frame display more into a parallel pipeline so that model calculations can take place on other CPUs at the same time as frame display. Thus, while frame display would take the same amount of time, other activities would not be waiting on its completion.

Frame read is the next highest part of the overhead. Depending on the size of the frames, the distance to the source and the speed of the connection, this overhead could grow or shrink. It depends on the application. When large data volumes must be moved around, it might be wise to consider network hardware intended for real-time simulation and process control. One network of this type is SCRAMnet (Shared Common RAM Network), a product of Systran (<http://www.systran.com>). Systran is a supplier of specialized, high-performance, high-availability computer data communications products for real-time applications. One version of their network is capable of 250 nSec/node transport delay and 16.7 MB/sec data throughput. Here again is an opportunity to use parallel pipeline software architectures. A thread could receive the next frame while the previous frame is being stored.

Inter-machine communications was surprisingly a minor part of the overhead. The fact that only two machines were involved contributed to this result. The experiments with increasing numbers of processors on a single machine confirms that adding processing devices increases communications overhead. If something like a SCRAMnet were employed for frame reception, a simple extension to that network could be made to facilitate inter-machine communications. There are always things that can be done to a LAN to improve the speed of connections. These are well known and not a research issue. It really just comes down to how much money you want to spend. The results to date show how to make best use of existing infrastructure. A more clever use of parallel-pipeline software architectures would improve these results. This is something that can be done without touching the hardware.

Frame storage to local devices could be improved by a faster local data bus. This would require replacing the motherboard of the cluster's host computer. Improvements here could also be had from faster storage devices.

It might be possible to convert  $\exp(x)$  to table lookup since each frame is composed only of integers. This is something that would be worth exploring. (Suggestion from Chris Mulliss, Ball Corporation.)

A question that should be asked is, "What overhead is added when one goes from one processor to multiple processors and from there to multiple machines? Is it really worth it to introduce the added complexity?" These experiments show that it is indeed worth it if improved processing throughput is a requirement.

Figure 51 shows the increasingly diminished execution time as additional processors are added. The one-processor cases show that there is little additional overhead due to the software required to manage multiple processors and multiple machines. Figure 52 shows that an improvement of 4-times the single-processor case was achieved. It must be admitted that this success came at the expense of dropping to 50% efficiency, as shown in Figure 53. However, the use of pipelines has the potential of greatly improving this situation.

Comparing Execution Times

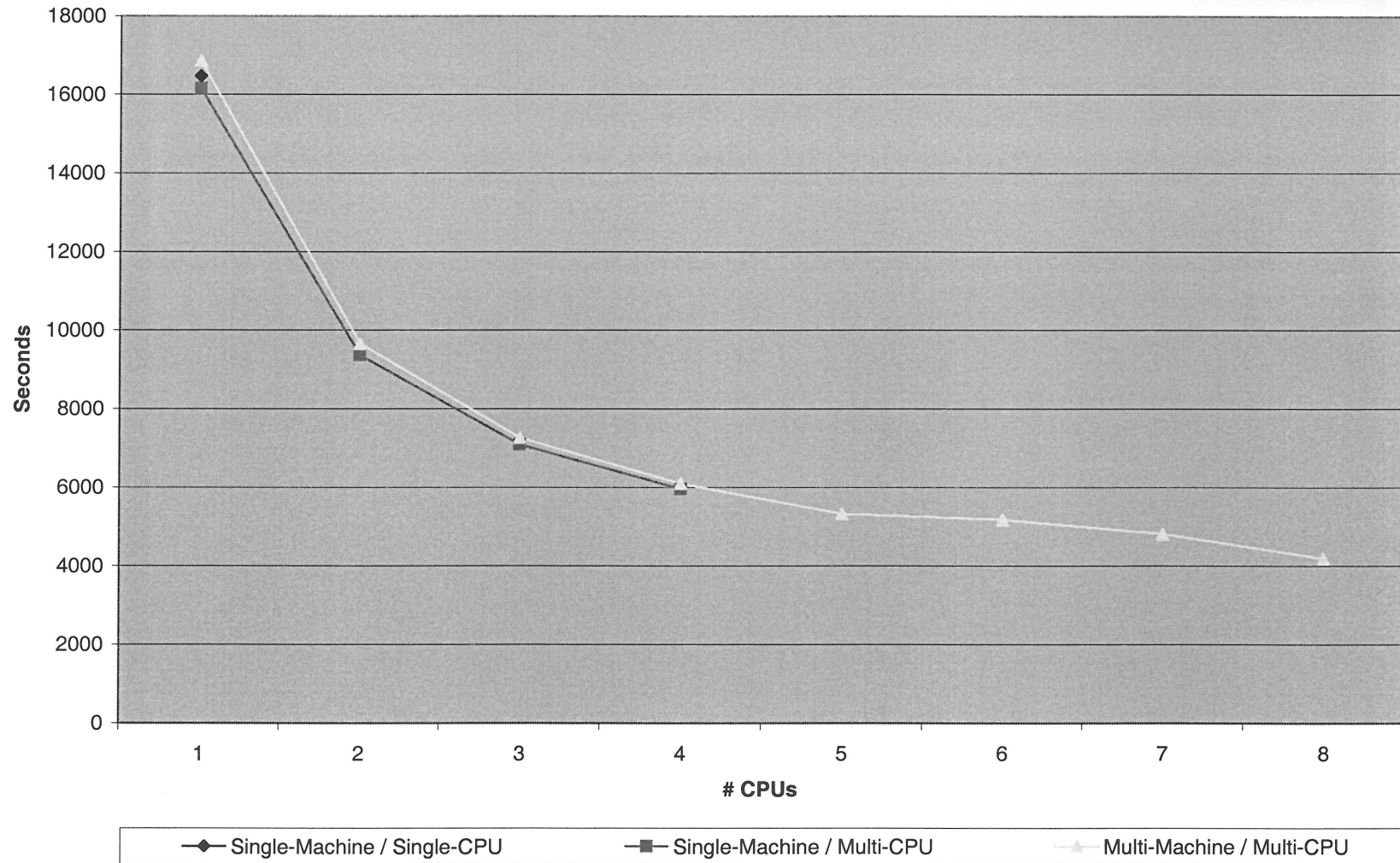


Figure 51. Comparing Execution Times

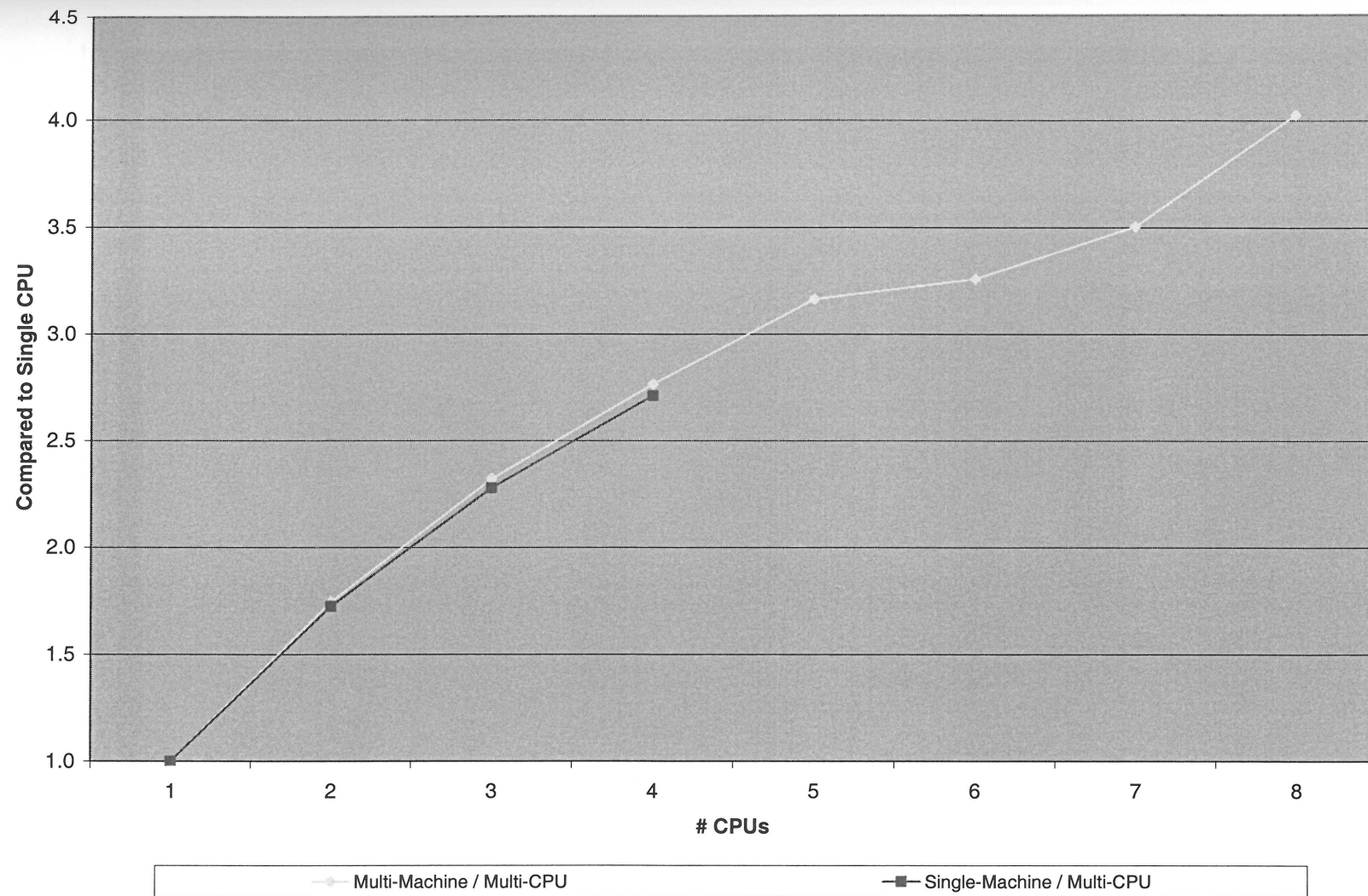


Figure 52. Comparing Speed-Up

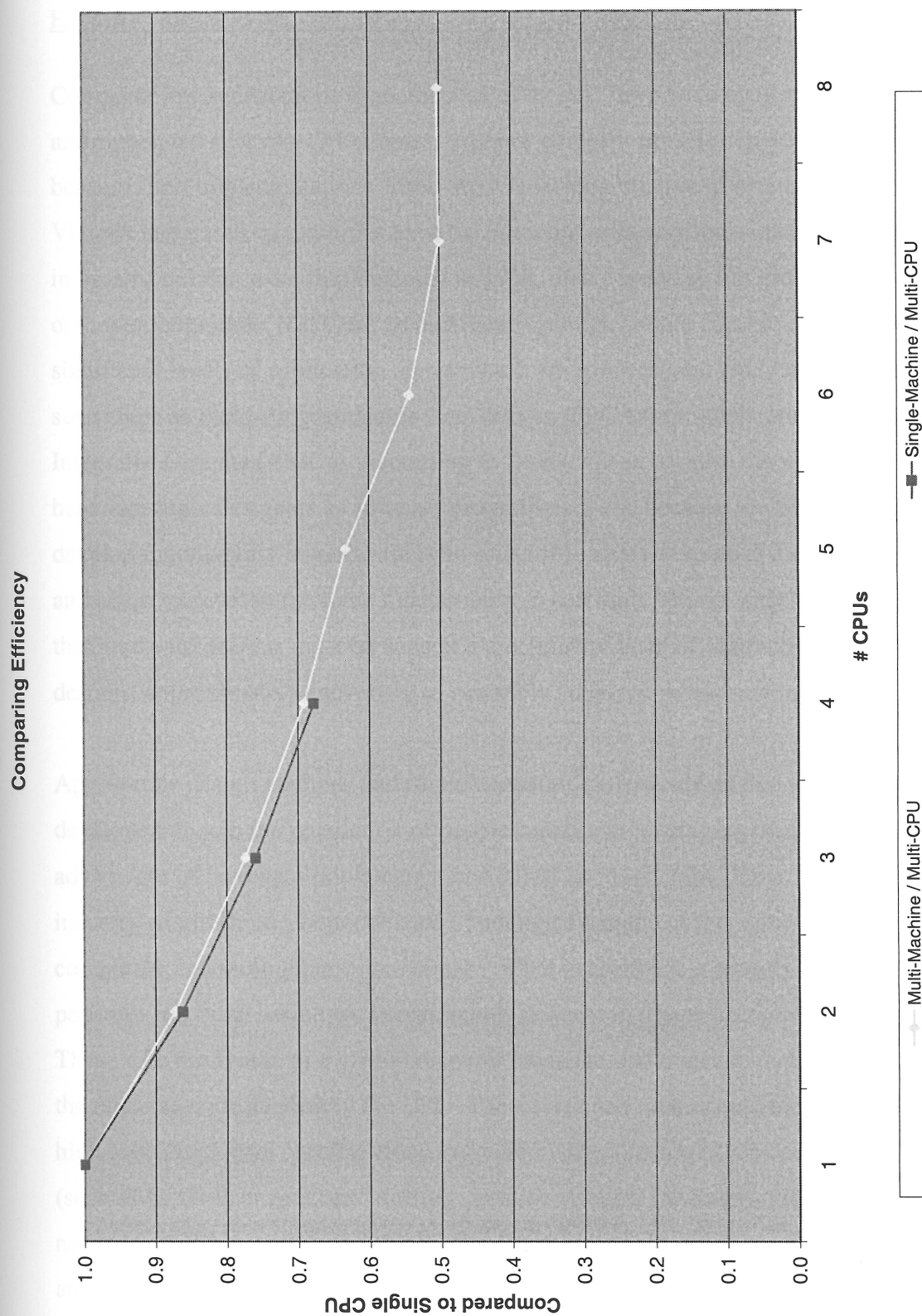


Figure 53. Comparing Efficiency



### Exploring Issues in Moving Toward FPGA Implementation

Computer implementations of mathematical models have wide application in simulation and embedded systems. Often times, superior complex models can not be employed because their implementations' throughput is lacking relative to practical requirements. Various integrated circuit types have the potential to relieve this situation. Since integrated circuits were first invented in 1958, their capability has grown by over eight orders of magnitude. Yet there are not nearly enough people capable of supporting significant levels of application conversion from conventional fetch/execute CPUs to such chips as Field-Programmable Gate Arrays (FPGAs) or Application-Specific Integrated Circuits (ASICs). According to Davis, this is because there are not as many hardware logic designers as software programmers and because the time and effort to develop functionality in hardware is an order of magnitude more difficult than the time and effort to develop the same functionality in software. This is largely due to the fact that hardware designs must be done at a much lower level of abstraction than software designs, approximately equivalent to assembly language programming.

Appropriate design methods and implementation tools would enable traditional software developers to join the population of people capable of leveraging the performance advantages of hardware implementations based on these chips. Then, all facets of industry would stand a better chance of taking advantage of this option for high-speed computing and throughput improvement. What is needed is a process and tool set that performs the same basic task as common high-level language compilers and linkers. These take the syntax of a given computer language and translate that syntax directly into the machine code needed by the CPU. The envisioned process and tool set would take high-level functional specifications expressed with commonly-used software languages (such as C, C++, or Java) and directly produce efficient hardware-oriented translations needed by FPGAs and ASICs. Given the current state of the art, producing such a process and tool set is a highly challenging strategic task. Furthermore, there is a change in mind-set that the typical software applications developer has to undergo in order to target such chips.

Xilinx has marketed a language compiler called Forge. This tool converts Java source code to Register Transfer Level (RTL) Verilog which can be synthesized, using established commercial tools, to FPGA implementations. Thus, it is possible that a program running under the standard Java runtime environment could be converted for FPGA use. This section discusses a preliminary effort to produce an FPGA implementation of a C++ engineering application using the Forge compiler. It includes lessons learned and clear-cut advice for others making this transition.

### *Background*

In an effort to improve throughput in a way that could ultimately result in frame-by-frame real-time, this project stepped completely out of the box of standard processing equipment. Field-Programmable Gate Arrays (FPGAs) are designed for high-speed computation. They are employed, for instance, in digital television at much higher frame-rates and resolutions than currently being used by this research. Therefore, they have the potential to achieve the throughput the Predictive Anomaly Detector (PAD) developed by this research will require for commercial application beyond after-collection post-processing. (For more technical details on FPGAs see Sharma (1998) and Oldfield & Dorf (1995). However, these details were not essential to the present research.)

At issue with FPGAs is that they are traditionally programmed from a chip-design perspective. This approach emphasizes gate-level constructs and register-level operations. PAD was built using a high-level language (C++) as a software application running on a generic CPU. Therefore, there is a very wide gap between PAD's original form and the form it would have to be in to run on an FPGA. Hardware description languages (HDL) exist that take a step toward simplifying the transition. The most popular of these in the commercial world is Verilog (Moorby and Thomas, 1998). Government applications lean toward the Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL, see Yalamanchili, 1997). A discussion of such languages on a more generic

plane is offered by Kloos and Cerny (1997).

There are higher-level languages that go farther than HDL or VHSIC. These make FPGAs more accessible to those who have skills in software but these languages are still based on the chip-design paradigm. An example of such a specialized language is Hardware Promela (Wenban and Brown, 1996). Another approach along these lines is a specialized C compiler (Data-Parallel Bit C, dbC) and architecture generator (Malleable Architecture Generator, MARGE) from Gokhale and Gomersall (1997). A follow-on work by Gokhale, Stone, Frigo, and Ahrens (2002) is StreamC. JHDL (Java Hardware Description Language) by Hutchings, Bellows, Hawkins, Hemmert, Nelson, and Rytting is another example of a using a software language to describe hardware at a low level.

These are followed by SA-C, a specialized language for graphics applications from Rinker, Hammes, Najjar, Bohm, and Draper (2000). Isshiki and Dai (1996) take the approach of using library extensions to GNU-compiled C++ to describe an FPGA design. However, their tool is specifically for hardware designers, not software developers. They cite several other approaches that all use non-generic dedicated compilers. Wo and Forward (1994) developed a partial-C compiler that takes generic code and generates the gate-level information needed by an FPGA. This is a step in the right direction but it does not directly accommodate object-oriented code.

What is needed is a process and tool set that performs the same basic task as common high-level language compilers and linkers. These take the application syntax and translate it directly into the machine code needed by the CPU. The envisioned process and tool set would take the high-level object-oriented syntax of an off-shelf language and directly produce the gate connections and register operations needed by the FPGA. Given the current state of the art in FPGAs, producing such a process and tool set is a mammoth strategic task. Still, more than ten years ago Patterson (1990) found that FPGAs are fast enough and large enough to accommodate the full range of modern applications. So, the dream of running a common software application on one or more FPGAs is not far fetched. Such enthusiasm must be tempered by Wo and Forward's (1994) caution that



modern bloat-ware will not cost-effectively fit on a pure FPGA system. Rather, they advise profiling the software and only placing the CPU-intensive parts on an FPGA co-processor. Knittel (1996) has demonstrated that FPGA co-processing on generic PC systems is readily achievable using off-shelf components and boards. (PC systems are the target of this present work. Designing a co-processor board for other system types should not be any more difficult.)

Xilinx is one of the world's largest manufacturers of FPGAs. They have undertaken to build Forge, a tool that translates Java source code into Verilog. Other tools already available from Xilinx translate the Verilog for use by simulators or directly by FPGAs. They can also translate for Application-Specific Integrated Circuits (ASIC). (ASICs are essentially write-once programmable devices built from the silicon up. FPGAs are slower but write-many devices that are continuously reconfigurable.) According to Davis (2002), the fact that FPGAs are immediately reconfigurable (as opposed to needing a prolonged manufacturing cycle as ASICs do) is important because it allows software designers to use a familiar "code, compile, debug" paradigm in an iterative fashion.

For this first effort, the author joined Xilinx' Forge beta test team and generated code that Forge translated directly into Verilog. Due to cost constraints, an attempt at actual FPGA insertion was not attempted. That is something that is left to future work.

### *Importance*

For many industrial applications the image frame size is large enough that PAD's underlying prediction model would contain 9,216,000 floating-point terms that would have to be computed and updated at least every 33 milliseconds. Achieving this objective is important to the technique being accepted for real-time operational use. Previous efforts with generic processors have not come close to achieving this throughput. Therefore, the author stepped completely out of the box and investigated FPGAs, programmable devices that one works with from a completely different point of view

compared to generic processors. Figure 14 places FPGAs in context with other options for improving throughput.

Digital Signal Processing (DSP) chips are very similar to generic CPUs. The difference is that DSPs are specifically architected for mathematical calculations. They are not designed for office automation as are generic CPUs. Because DSPs are similar to generic CPUs, converting PAD to DSPs would not be as big a leap as converting to FPGAs.

Significant speed improvement could still be achieved thereby since DSPs are commonly used for real-time implementations that require intense math calculations. However, they are not as big a step out of the box as the author wanted to take. The risk would be lower but that would not allow exploration of the cutting edge in transitioning common software applications to high-speed chips. In addition, Davis (2002) indicates that FPGAs provide a much better platform for implementing algorithms that can be highly parallelized at a fine level of granularity and for implementing algorithms that can be heavily pipelined (that is increasing overall throughput at the expense of latency.)

Further, Bhatia (1997) found that FPGAs break the fetch/execute paradigm of traditional processing approaches with the effect that flexibility and performance are expanded at the same time. He shows a figure (repeated in Figure 54) that demonstrates the typical curve that compares instruction set architectures to custom hardware. The former gives high flexibility but relatively low performance whereas the latter gives high performance but low flexibility. FPGAs have the potential for both.

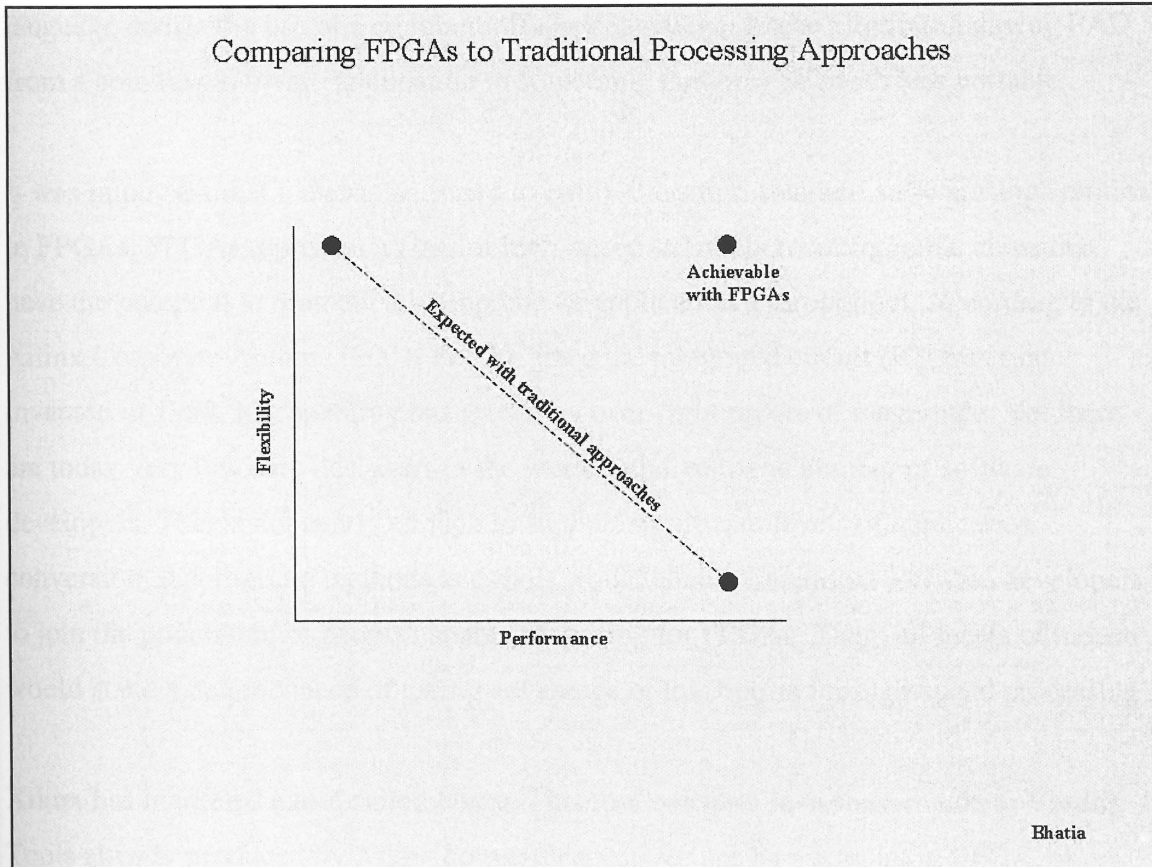


Figure 54. Comparing FPGAs to traditional processing approaches

### Approach

Having decided to transition PAD to running on FPGAs, the next step was to decide on a general approach. There are some tools available to assist in the process of converting standard software applications to FPGAs. Most of these require converting to integer calculations, the use of specialized languages that begin as extensions to standard languages, or the redesign of the application in a form compatible with a hardware description language. The latter approach essentially requires that the application be recast as a gate-level and register-level design. This is a rather onerous task for a complex application since it requires both reimplementing the desired functionality while simultaneously specifying the low level gate implementation. As such, it is a task more suited to chip designers rather than software applications developers. Using a specialized

language denies the use of a common off-shelf language. It also requires changing PAD from a common software application to something that may be much less portable.

It was important that a means be found to easily transition standard software applications to FPGAs. FPGAs represent a class of high-speed infinitely reconfigurable chips that have the potential to dramatically improve an application's throughput. According to the Xilinx Corporate Authors (2001, Feb 8), since the integrated circuit (IC) was first invented in 1958, IC capability has grown by over eight orders of magnitude. Yet there are today very few chip designers in the world, relative to the number of software developers. This is not nearly enough to support significant levels of application conversion. Appropriate methods and tools would enable traditional software developers to join the population of people capable of writing for FPGAs. Then, all facets of industry would stand a better chance of taking advantage of this option for high-speed processing.

Xilinx has marketed a tool called Forge. This tool converts Java source code to Verilog. Tools already produced by Xilinx convert Verilog for use by a simulator, FPGA, or ASIC. Thus, it is possible that a program running under the standard Java runtime system (Java Virtual Machine, JVM) could be directly converted for an FPGA. (Davis (2002) has said that Xilinx has plans for a C-syntax front-end for Forge. They currently have the basic operations, decisions and loops supported and will be working on pointers and arrays next. C++ is still not on the schedule but once they get C completed they will take a look at it.)

To begin the process of porting the computationally intense parts of PAD from standard C++ to Verilog. The following steps were followed:

1. translate appropriate C++ code to Java
2. ensure that part of the model executes correctly in a standard Java environment
3. use a commercial tool (Forge) to translate Java source code to Verilog
4. actively participating on Xilinx' Forge beta test team

Participating on the Forge beta test team and providing input to Forge's development was a large part of the project. Forge represents an important strategic technology that is well worth exploring and fostering.

The general approach taken by this project for moving from software application toward FPGA implementation started with a fully operational software application. Once the program was performing as desired, the Java source code was fed into Xilinx' Forge, along with preferences and constraints to guide Forge's operation. Forge analyzed the source code and generated resistor-transfer logic (RTL) hardware description language (HDL) in the form of Verilog. The Verilog code is in standard form and can be used directly by FPGA synthesis or simulation tools. Physical implementation on an FPGA does not necessarily require building a specific circuit board since there are numerous standard boards available.

The prototype for this part of the work was the Java version of PAD. Testing compared the Java version of PAD with the C++ version using the voice/fan data (Figure 9).

### *A Simple Java/Forge Case*

Just to get a sense of the vast difference between a high-level language like Java and a low-level language like Verilog, have a look at the following example. Java allows a much more direct means of approaching a programming problem whereas Verilog requires thinking at the gate and register levels, a more detailed paradigm than working in assembler.

After installing Forge, the following very simple Java program was written and processed with that tool to produce the Verilog equivalent:

```

public class Main extends Object
{
    public Main ()
    {
    }

    public static void main (String args[])
    {
        int i, j;

        for(i = 0; i < 20; i++)
        {
            j = i * 3;
            j = j + 5;
        }
    }
}

```

On the following two pages is the Verilog equivalent of the above code as produced by Forge. As one can see, it takes a lot of obscure Verilog code to perform a simple Java process. Therein lies the main barrier to employing high-speed FPGA chips, even if their throughput potential is immense. That is why one would want to just feed the Java code into a "compiler" and have the Verilog produced automatically. This is very similar to what a traditional compiler does for fetch/execute CPUs. Davis (2002) asserts that while it is true that the Verilog is much more obscure and at a lower level of abstraction when compared to the Java code, the Verilog code that Forge generates is not optimally coded for human readability (although it does generate quite good hardware implementations ultimately). That is, if a good Verilog designer came along and tried to replicate the same functionality by going back to first principles, it is likely that the Verilog code they wrote would be much simpler.

```

// Standard Verilog translation of Main.main
App([Ljava.lang.String;)
// Tue Jul 24 19:51:12 EDT 2001

module L_Main_main_App_$L_java_lang_String (
    DONE,
    ARG0,
    GO,
    RESET,
    CLK);
input    CLK;
input    RESET;
input    GO;
input    [31:0]    ARG0;
output    DONE;
wire      Main_main_[Ljava.lang.String;5857105_1_done0;
wire      muxEntry_15_result;
wire      [31:0]    muxEntry_18_result;
wire      muxEntry_20_done;
wire      muxEntry_20_result;
wire      emux_22_result;
// #-1:    <unknown source for line# -1>
L_Main_main_$L_java_lang_String
Main_main_[Ljava.lang.String;5857105_1_instance(.CLK(CLK),
.RESET(RESET), .GO(muxEntry_15_result),
.ARG0(muxEntry_18_result),
.DONE(Main_main_[Ljava.lang.String;5857105_1_done0)));
// UN-arbitrated mux
assign    muxEntry_15_result = GO;
// UN-arbitrated mux
assign    muxEntry_18_result = ARG0;
// UN-arbitrated mux
assign    muxEntry_20_result =
Main_main_[Ljava.lang.String;5857105_1_done0;
assign muxEntry_20_done =
Main_main_[Ljava.lang.String;5857105_1_done0;
assign emux_22_result = (muxEntry_20_done == 0) ?
muxEntry_20_result : muxEntry_20_result;
assign    DONE = emux_22_result;
endmodule // L_Main_main_App_$L_java_lang_String_0

```

```

module L_Main_main_$L_java_lang_String (
    DONE,
    ARG0,
    GO,
    RESET,
    CLK);
input      CLK;
input      RESET;
input      GO;
input      [31:0]    ARG0;
output     DONE;
wire       [31:0]    iinc_13_4;
wire       if_icmplt_19_6_true;
wire       if_icmplt_19_6_false;
wire       if_icmplt_19_6_compare;
wire       and_7;
wire       [31:0]    decoder_8;
wire       or_9;
reg        [31:0]    enabled_register_10;
reg        register_11;
wire       and_12;
// #12:     for(i = 0; i < 20; i++)
assign     iinc_13_4 = (decoder_8 + 32'h1);
// #12:     for(i = 0; i < 20; i++)
assign     if_icmplt_19_6_true = if_icmplt_19_6_compare;
assign     if_icmplt_19_6_false = !if_icmplt_19_6_compare;
assign     if_icmplt_19_6_compare = (!(decoder_8[31] ^ 1'b0) &&
        (decoder_8 < 6'h14)) ||
        ((decoder_8[31] ^ 1'b0) && (decoder_8[31]));
assign     and_7 = or_9 & if_icmplt_19_6_false;
assign     decoder_8 = (enabled_register_10[31:0] &
        {32{register_11}}) | (32'h0 & {32{GO}});
assign     or_9 = register_11 | GO;
assign     and_12 = if_icmplt_19_6_true & or_9;
assign     DONE = and_7;
always     @(posedge RESET or posedge CLK)
begin
    if (RESET)
        begin
            enabled_register_10 <= 0;
            register_11 <= 0;
        end
    else
        begin
            if (and_12) enabled_register_10 <= iinc_13_4[31:0];
            register_11 <= and_12;
        end
    end
end
endmodule // L_Main_main_$L_java_lang_String_2

```



### *Translating PAD/C++ to PAD/Java*

At first it was thought that the translation of the original PAD/C++ to PAD/Java would not take long. It turns out that this was very time consuming. PAD/C++ is built as a Visual C++ GUI application. Also, the control code is tightly integrated with the display code. It was necessary to separate the two in order to avoid having to translate the GUI. This is one of the architectural issues to keep in mind when writing code that is intended for FPGA use. In the architecture envisioned here the FPGA would act as a co-processor performing the math-intensive part of the code in cooperation with the control and GUI modules. The control module would feed data to the GUI which would operate independently of PAD.

Other translation issues were encountered:

- Java's way of performing text and binary file I/O is more flexible than C++ but also more cumbersome.
- Java requires the inclusion of numerous try/catch pairs.
- Java does not support certain variable types such as "unsigned long int".
- Java does not support structures. These all need to be converted to classes.

The C++/Java translation issues were managed via careful attention to detail and thorough testing. Some data structures and memory allocations were modified to accommodate Java's way of doing things. In the end, there was no functional difference between PAD/C++ and PAD/Java.

### *Issues Encountered While Feeding Java Code to Forge*

Once Forge was installed and its operation understood, PAD/Java was processed. Two issues were encountered: Forge does not directly support floating point variables. It only supports integers. It is possible to force an FPGA to perform floating point operations directly but the cost is unacceptable in terms of chip space and throughput. FPGAs are

inherently integer devices. So, it is worth looking into integer representations of floating point values, and the attendant operations. PAD makes heavy use of floating point representations of floating point values so this is a considerable shift in programming style. According to Davis (2002) some Xilinx customers claim that the use of floating point within an FPGA still provides significant advantages over general purpose processors. The key is that these customers have developed or acquired highly optimized/pipelined floating point proprietary libraries that Forge calls directly.

The other issue is that if the required chip interface is more complex than can be represented with a method call paradigm, then Forge requires the use of a special I/O package to describe the pins and their clock timing relationships. This package is included with the Forge binaries.

### Casting Floating Point Values as Integer Variables

The shift from floating point variables to integer variables is rather large if one has never worked in pure integer. This project explored information sources and published literature. It also examined several publicly available libraries. Only one library, MathFP by Hommes (2001), proved suitable for PAD application. This is mainly because it is the only one that implements  $\exp(x)$ , the basis function used to build PAD's prediction model. Also, it appears to have the best numerical accuracy overall, although a formal study was not conducted. Unfortunately, MathFP is publicly available only as a Java class file. Source code is not distributed. The documentation does not expose its underlying methods. Tests with MathFP are described in the next section.

An excellent introduction to the topic of integer (fixed-point) math is given by Lemieux (2001). He shows how floating point values can be translated to integer variables while retaining arbitrary decimal accuracy. He then describes the basic operations: add, subtract, multiply, and divide. Given this information, one might easily jump to the conclusion that Taylor's Series (Shanks and Gambill, 1969, pp 195-212) could be used to generate functions such as  $\exp(x)$ . However, experience with the representations and

operations described by Lemieux (2001) showed that the denominator of the Series grows far too quickly for the accuracy required by PAD to be achieved. Due to the resulting numeric overflow, sufficient numbers of terms can not be calculated.

CORDIC (COordinate Rotation DIgital Computer) is a popular and effective method for performing fixed-point math. This method was originally a special purpose computer for aircraft navigation. It has since come to stand for the method embodied in that computer. According to Jones (1992), the algorithm is based on coordinate rotation. An initial vector  $P_0$  is rotated until some condition is met. The vector  $P_0$  is determined by the coordinate system,  $m$ , and two initial points,  $x_0$  and  $y_0$ . The coordinate systems are: circular ( $m = 1$ ), linear ( $m = 0$ ), and hyperbolic ( $m = -1$ ). He goes on to say that the advantage of this method over more traditional approaches like power series include:

- The same algorithm can be used to calculate multiplication, division, sin, cos, tan, arctan, sinh, cosh, tanh, arctanh, ln, exp, and square-root.
- The algorithm uses only shifts and adds so it will usually be faster than an equivalent power series.
- Sin/Cos and Sinh/Cosh are calculated simultaneously.

Cylix (1997) presents an elementary tutorial and example on this topic. He says that almost all scientific calculators use the CORDIC technique and that it works by rotating the coordinate system through constant angles until the angle is reduced to zero. The angle offsets are selected such that the operations on points  $X$  and  $Y$  are only shifts and adds. Read the introduction by Jarvis (1990) after you absorb the one by Cylix (1997).

Knaust (1997) also gives an introduction, although his is more theoretical. He does reveal that CORDIC was originally developed by Volder (1959). Figure 55 is from Knaust's (1997) discussion. It shows how the coordinate system is rotated from its original configuration (red point 0) via smaller and smaller steps to its final point where the angle between the new axis and its intended position is essentially zero. Eklund is similarly theoretical but he gives more in the way of historical citations.

Andraka (1998) writes a very understandable and useful survey of functions implemented via CORDIC. He discusses the underlying theory and then extends it to specific functions. He goes on to implement his material in FPGAs. In this paper and others it is interesting to notice the difference between the way a hardware designer approaches “programming” an FPGA compared to that of a software applications developer. The hardware designer is concerned with wiring gates, adders, and registers whereas the software developer is concerned with more abstract logic, essentially at the level of mathematical symbology. The two meet in the middle since the FPGA requires (for all intents and purposes) pure integer programming and CORDIC is designed for direct hardware implementation on integer devices. One also sees CORDIC used frequently on generic CPUs to lend additional speed to graphical applications, particularly in the online gaming market.

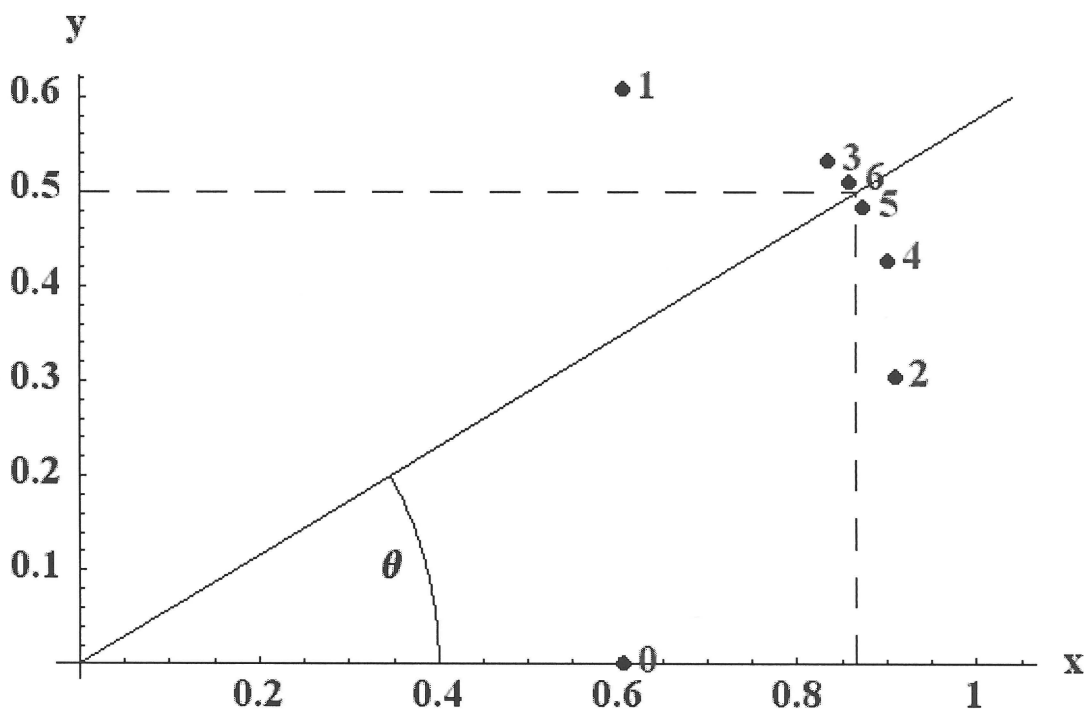


Figure 55. Example of CORDIC coordinate system rotation (Knaust)

## Testing the MathFP Java Fixed-Point Library

Hommes' (2001) library for manipulating floating-point values as integers proved quite satisfying to work with. Tests were conducted with Forge to judge MathFP's compatibility with that tool. (MathFP was never written with Forge or FPGAs in mind.) Tests of were also conducted to judge MathFP's accuracy in conducting basic operations required by PAD: iterations and  $\exp(x)$ . No statistically significant differences were found between MathFP and floating-point Java while running iteration and  $\exp(x)$  tests.

### *Test of MathFP Iterations*

MathFP was used to perform a floating point iteration. An integer variable was iterated from  $-1.5$  through  $1.5$  in steps of  $0.05$  using the following code. A comparison of the integer iteration with the normal floating-point iteration showed that there is no statistically significant difference between the two.

```
public static void main (String args[])
{
    int increment = MathFP.toFP("0.05");
    int result = MathFP.toFP("-1.5") - increment;
    double testValueInc = 0.05;
    double testValue = -1.5 - testValueInc;
    Double stringNumber = new Double(0.0);
    double resultFloatingPoint;
    double percentDiff;
    String thisString;

    // Incrementation test
    for(int i = 0; i < 61; i++)
    {
        result = result + increment;
        testValue = testValue + testValueInc;
        thisString = MathFP.toString(result);
        stringNumber = stringNumber.valueOf(thisString);
        resultFloatingPoint = stringNumber.doubleValue();
        percentDiff = 100.0 * ((testValue - resultFloatingPoint) /
                               testValue);
        System.out.println("result =," + resultFloatingPoint + "," +
                           + testValue + "," + percentDiff + ",%");
    }
}
```

t-Test: Two-Sample Assuming Equal Variances		
	<i>MathFP</i>	<i>Java Double</i>
Mean	0.001459016	1.45967E-15
Variance	0.789384184	0.787916667
Observations	61	61
Pooled Variance	0.788650425	
Hypothesized Mean Difference	0	
df	120	
t Stat	0.009073353	
P(T<=t) one-tail	0.496387839	
t Critical one-tail	1.657649591	
P(T<=t) two-tail	0.992775678	
t Critical two-tail	1.979929038	

#### *Test of MathFP Computing Exp(x)*

MathFP was used to insert the interacted value computed above as the argument in  $\exp(x)$ . An integer variable was iterated from  $-1.5$  through  $1.5$  in steps of  $0.05$  and used as input to  $\exp(x)$  using the following code. An comparison of the integer version with the normal floating point version showed that there is no statistically significant difference between the two.

```

public static void main (String args[])
{
    int increment = MathFP.toFP("0.05");
    int result = MathFP.toFP("-1.5") - increment;
    double testValueInc = 0.05;
    double testValue = -1.5 - testValueInc;
    Double stringNumber = new Double(0.0);
    double resultFloatingPoint;
    double percentDiff;
    String thisString;

    // exp(x) test
    for(int i = 0; i < 61; i++)
    {
        result = result + increment;
        testValue = testValue + testValueInc;
        thisString = MathFP.toString(MathFP.exp(result));
        stringNumber = stringNumber.valueOf(thisString);
        resultFloatingPoint = stringNumber.doubleValue();
        percentDiff = 100.0 * (Math.exp(testValue) -
                               resultFloatingPoint) / Math.exp(testValue);
        System.out.println("result =," + resultFloatingPoint + "," +
                           Math.exp(testValue) + "," + percentDiff +
                           ",%");
    }
}

```

t-Test: Two-Sample Assuming Equal Variances		
	<i>MathFP</i>	<i>Java</i>
Mean	1.437734426	1.435103788
Variance	1.423322213	1.416020024
Observations	61	61
Pooled Variance	1.419671119	
Hypothesized Mean Difference	0	
df	120	
t Stat	0.012193185	
P(T<=t) one-tail	0.495145868	
t Critical one-tail	1.657649591	
P(T<=t) two-tail	0.990291736	
t Critical two-tail	1.979929038	

### *Summary and Recommendations*

Parallel processing is one way of improving a software application's throughput and it extends the fetch/execute style of computing using standard CPUs. A step beyond that is the use of digital signal processing (DSP) chips. These are similar to standard CPUs but are architected for efficient math operations. They are designed for engineering applications rather than business applications. This project stepped completely away from fetch/execute means of computing and examined Field-Programmable Gate Arrays (FPGAs).

In its present state, Forge assumes that the source code is compatible with the Sun standard Java compiler. Since the author's original model is written in C++ it was necessary to translate the compute-bound part of the model to Java. This was no simple task. Many have said that there is a strong similarity between Java and C++. However, at the detailed level of syntax translation there are many issues that had to be reconciled.

The transition of a conventional software application from floating-point to fixed-point<sup>®</sup> operations and variables is also not trivial. One needs a fixed-point library of reasonable precision to accommodate this. An attempt to build a library of arithmetic operations and then to use those to create a Taylor's Series for  $\exp(x)$  failed due to word-overflow problems. The denominator grows too quickly for reasonable precision to be achieved. This author did find a public-domain Java class file that provided the necessary arithmetic and other functions at the required accuracy. Tests showed that the Java version of PAD and that the Java fixed-point library produce results that are not statistically different from the original C++ code. The next step would be to synthesize the Verilog code for FPGA insertion. Davis (2002) expects that future versions of Forge will provide more sophisticated methods for helping with the translation from floating point to integer implementations and that as FPGAs continue to grow in size and capability, Forge will also provide ways to implement floating point directly in the FPGA.



Given these issues, why would anyone want to go through the learning process and effort needed to transition conventional software to FPGAs?

- There is a positive impact on time-to-market when functionality can be developed and tested with high-level languages without the additional overhead of hardware or hardware simulators. These languages facilitate thinking and testing at a more abstract level, closer to the application's purpose.
- There are relatively few people in the world today able to write effective code directly for an FPGA. High-level-language compilation for FPGAs would greatly expand the number of people capable of building applications for these chips.
- Processes and tools for going directly from conventional software to an FPGA would give a tremendous boost to throughput, physical size, and dual-platform operation.
- FPGAs give a speed boost that is well beyond that achievable with generic CPUs and DSPs. Along with that speed they give a flexibility that is not available from custom hardware. The cost incurred for the speed is far less than one would experience with custom hardware or ASICs. One also has to compare the complexity and the attendant additional costs of achieving the required speed with multiple CPUs and networked machines, although this approach may give comparable flexibility. Still, multiple CPUs do not scale linearly, especially for algorithms that can make use of fine-grained parallelism.
- The standard US Air Force software cost model (REVIC), <http://www.asu.faa.gov/ASU200/Toolbox/asu200afa.htm>, <http://www.jsc.nasa.gov/bu2/resources.html>) is based on actual experience with defense-related software projects. It clearly shows that there is a direct relationship between the number of lines of executable code and the cost of the project. (Certain project environmental factors determine the slope of cost growth as the lines of executable code increases.) In the case of our simple example, the Java version had

only one-half page of code, whereas the Verilog version had 1.5 pages of code. This besides the fact that Java is a high-level language that is much easier to work with than the very low-level Verilog. Larger designs should demonstrate even greater increases in code size, typically, in Davis' (2002) experience, about 5 to 10 times larger. Thus, the cost of developing directly in Verilog for the same application is likely to be much larger than for a language like Java.

Clearly though, there are some paradigm shifts in software development that have to take place. The largest of these is programming directly in fixed-point rather than staying with floating-point. This is something Xilinx is currently working on. Their goal is to be able to provide a functional implementation for any code that uses floating point so that designers can at least get a baseline implementation. Then they can decide if they want to re-implement using fixed point to get better performance. For now though, the application has to be written from the ground up with the target hardware (FPGAs) in mind, although the resulting code will still execute on any machine running the standard Sun JVM. The advantages of high-level languages are still available but new skills have to be learned. Additionally, Forge requires a certain documented programming style so that the developer is not totally free in that choice. Finally, there are documented ways to write the code so that less space is used on the chip. These need to be kept in mind from a throughput perspective as well. In any case, a project that targets FPGAs could be written directly in Java given the current state of Forge. To the author's knowledge, there are no other tools that support object-oriented design and programming for FPGAs. It should also be recognized that to get the best performance out of any hardware requires writing the code with that hardware in mind.

Ultimately, tests with an FPGA simulator should be performed. Once satisfying results with the simulator are achieved, the next step is to acquire an FPGA prototype board and port the model to that board. These tasks require the use of simulator software and software that translates Verilog to the code actually placed on the chip. Prototype boards, simulators, and Verilog translators are all tools that would have to be purchased. Thus, they represent a milestone in the project that requires more justification than did the

## Chapter Five

### Conclusion, Implications, Recommendations, Summary

#### Summary

There were several major steps to this research:

1. find and implement a next-sample prediction capability
2. apply the predictor as input to a detector that separates anomalies from outliers and background
3. employ a series of data streams of varying robustness, while measuring the number of samples required to construct the model to achieve a given level of detection accuracy (fixed and moving anomalies)
4. develop and implement an objective means for estimating detection accuracy under the above conditions, estimate the detection accuracy of the model in terms of false alarms and good detects
5. show that the chosen technology is linearly scalable in terms of its mathematics, data flow, and automated-computation implementation

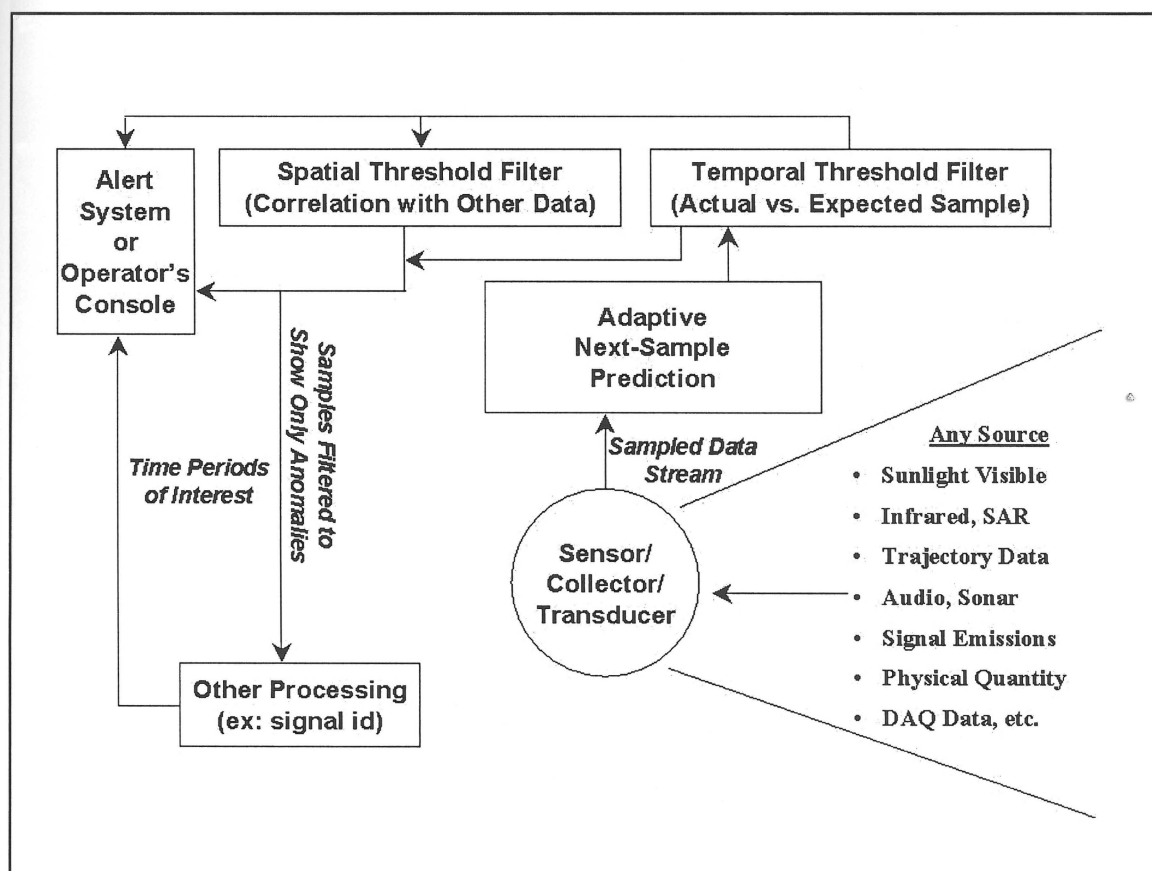
#### Conclusion

The following are the major outcomes of this research:

- The prediction error curve steadily decreases to below 20%.
- The gaussian predictor proved to be superior to the linear predictor.
- False alarms were less than 10% for the single- and two-dimensional data.
- Anomalies were found in all but the first pixel in which they exist.
- It was possible to detect anomalies for a substantial length of their existence.
- False alarms increase with the spectral content of the data, for a given model size.
- Computational complexity increases linearly with problem complexity.
- It was harder to find the end of an anomaly than the beginning.
- The parallel nature of the mathematics facilitate throughput improvements via fetch/execute CPU architectures and via programmable devices such as FPGAs.
- The technology developed by this research was found to be very general, having application in a wide range of domains using a variety of data streams.

## Implications

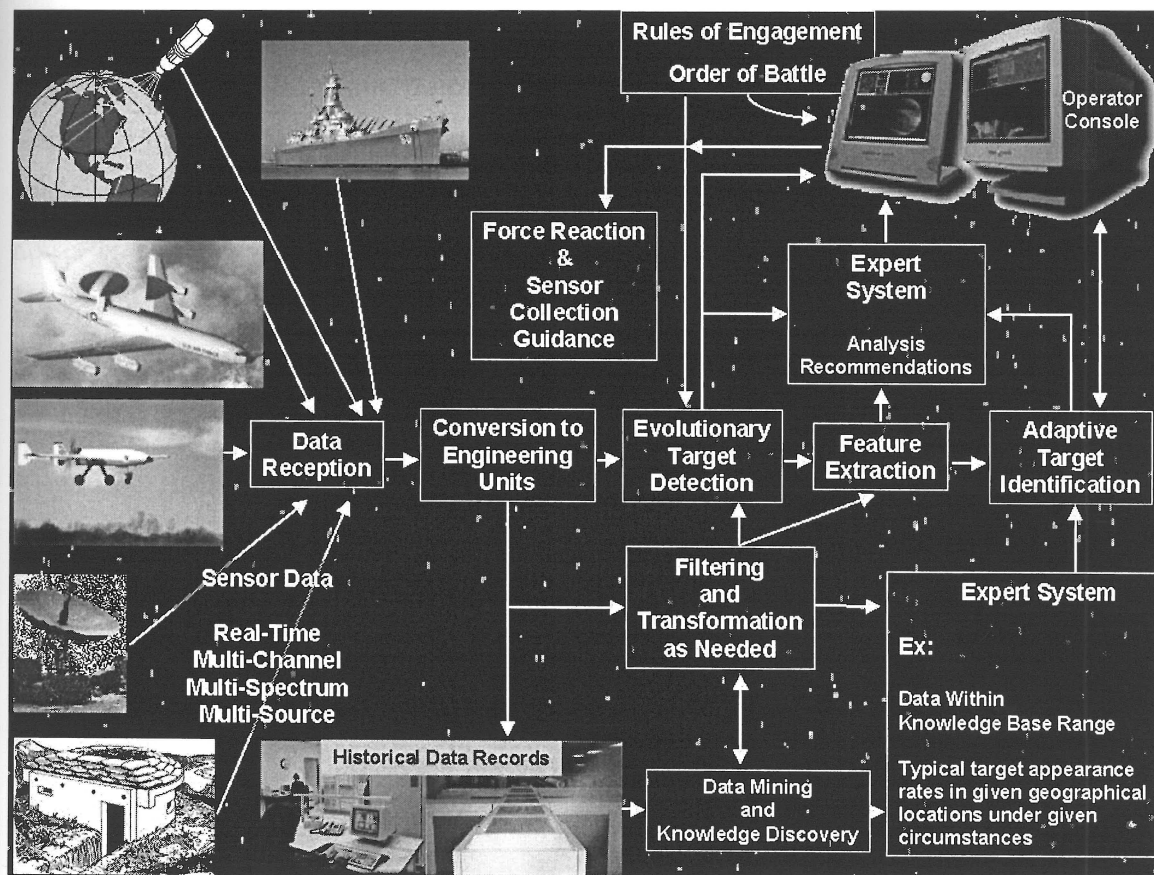
An architecture employing this method is shown in Figure 56. This research provides an effective predictive anomaly detector (PAD). The results can be posted for human viewing, evaluation, and reaction. These results can also be submitted for additional processing, such as signal classification. There are no limitations as to the source of the data, as long as the "staring, continuous-dwell" requirements are met, and there is a spectral limit to the data.



**Figure 56. Sample architecture for PAD employment**

This architecture can be employed on shipboard, airborne, or land-based platforms. There is a potential for supporting real-time and offline scenarios. From a national defense

perspective, a larger concept of operations using this architecture, to include force reaction and collection guidance, is illustrated in Figure 57.



**Figure 57. Hybrid adaptive systems for real-time sensor signal utilization**

This operations concept is based on the premise that adaptive algorithms can effectively find potential events of interest in dynamic environments. The value of PAD would be in those domains where normal threshold detection techniques and automatic gain control methods would not succeed in finding subtle anomalies. In essence, PAD enables applications to dynamically adjust to robust backgrounds and find low-observable events that would normally go undetected and, therefore, not tracked. This capability does not change the original system setup for the sensor since the technique is added to the existing signal processing capability as a real-time or post-collection enhancement.

PAD converges on a model of the background environment and maintains track of the background's dynamic change. When an event occurs within the sensor's field of view,

an adaptive algorithm determines if the return is sufficiently different from the background prediction. Domain-specific identification algorithms can then be applied to verify if the detection is of interest. This interaction between the detection and classification algorithms determines if the potential event is to be tracked or if it is a false alarm. An important aspect of the adaptive approach is a dynamic detection threshold that could enable these systems to find signals and events that would normally be lost in the noise. The adaptive algorithms thus provide additional decision information to the observation system that can enhance its ability to find events of interest in environments subject to many false alarms.

## Recommendations

The difficulties with the sine/cosine phase shift were unexpected. Only the beginning and end of the anomaly were found. Typically, the body of the anomaly is also detected. More reflection is needed on means of reconfiguring the detector so that it can find anomalies of this nature, involving only a phase shift in the background. From the test results, it appears as if one could compare the average and instantaneous RRPE. The instantaneous RRPE has a huge reaction to phase shifts on both ends of the anomaly. The evolving RRPE standard deviation also shows a definite reaction at either end of the anomaly. In both cases, the spike in instantaneous RRPE and the step in evolving standard deviation, could be used to identify the beginning and end of anomalies.

Other ways of preprocessing the data could be explored. For instance, does the spectral content shift when anomalies occur? Can the detection statistics be improved by observing various transformations of the data at the same time? In the case of sensor data stream, can isolating and simultaneously observing several planes of the stream improve detection statistics? For instance, in the case of the QuickCam, would it be more useful to observe the Red, Green, and Blue planes separately? The benefit would have to be balanced against the additional computational and data flow workload.

Are there other prediction methods that would enable greater or equal detection accuracy with less computation workload? Other non-linear prediction models could be explored. Additionally, one could explore means of automatically building a model that generates (heuristically) the data previously received. This is much different from a next-sample predictor and would be useful as a baseline when an anomaly is currently being detected. During such times, the predictor stops training and so is unreliable as a source of baseline comparisons.

So that the technology developed by this research is not seen as fit only for toy problems, the author employed data collected from real sensors under less than ideal conditions. The author also explored ways of using present technology to satisfy present needs for throughput, while laying the groundwork for satisfying future needs. In this way, it should be possible move applications at least one year ahead of the technology curve, rather than always leaving them one year behind. A strong effort was made in parallel/distributed processing on common networked engineering workstations. The author also joined Xilinx' Forge beta test team and generated Java code that Forge translated directly into Verilog. Due to cost constraints, an attempt at actual FPGA insertion was not attempted. This is a step that needs to be taken. Since having PAD cast<sup>®</sup> in standard fixed-point Java code is a prerequisite, a preliminary step could be to insert that code into a JStamp (<http://jstamp.systronix.com>). This is a hardware version of the standard Sun JVM. Going to an FPGA is a far more expensive step given the need for costly software and hardware. However, if the sponsorship can be arranged, it is an important step to take.

A related area of research is human factors. How should the results be displayed? How should the human interact with the detection process? Can the results display be used to support that interaction, via touch screen or mouse, for instance?

There are clustering techniques that could be developed. Spatially, should nearby pixels be said to have detections when a given pixel has a verified detection? Temporally, should neighboring samples be said to be a part of a given sample's verified detection?

When an anomaly is detected, the predictor stops evolving and its "output" is set equal to the moving average of the moving average of some number of pre-detection samples.

Would it be better to employ a data stream model instead of the moving average?



## Appendix A

### Mathematics Behind the Anomaly Detection Model's Prediction Phase

Gaussian radial basis functions are the foundation of the detection model's prediction phase. Basis functions are simple-equation building blocks that are a proven means of modeling more complex functions. Brown (in the book edited by Light, 1992, pp. 203-206) showed that if  $D$  is a compact subset of the  $k$ -dimensional region  $R^k$ , then every continuous real-valued function on  $D$  can be uniformly approximated by linear combinations of radial basis functions with centers in  $D$ . Proofs of this type have also been shown by Funahashi (1989); Girosi and Poggio (1989); and Hornik, Stinchcombe, and White (1989). The theory thus having already been well established, the author proposes to concentrate on practical implementation and application to unspecified anomaly detection in staring continuous-dwell sensor data streams. The term "point" refers to a location in the space  $R^k$ . The intended application has  $k = 1$ , a one-dimensional value such that  $R^k$  is a line. Look to Sanner and Slotine (1991, May/November; 1991, December) for an expansion to larger dimensions. This appendix interprets their work for one dimension.

At the top level, complex functions can be approximated using the following summation:

$f(x) = \sum_{i=1}^n [c_i g_i(x, \xi_i)]$	<p><math>f(x)</math> approximates function <math>F(x)</math> at point <math>x</math> in region <math>R^k</math></p> <p><math>\xi_i</math> is the center or location of basis function <math>i</math> in region <math>R^k</math></p> <p><math>g_i</math> is the nonlinear basis function <math>i</math> centered at <math>\xi_i</math></p> <p><math>c_i</math> is the weight by which the output of <math>g_i</math> is multiplied</p> <p><math>n</math> is the number of basis functions</p>
---	--

One picks the locations of  $n$ -number of basis functions in  $R^k$  and then chooses a function to use at each location. Set the output weight for each function and then add up the results of those functions at input point  $x$ . The interesting part, of course, is how to carry out

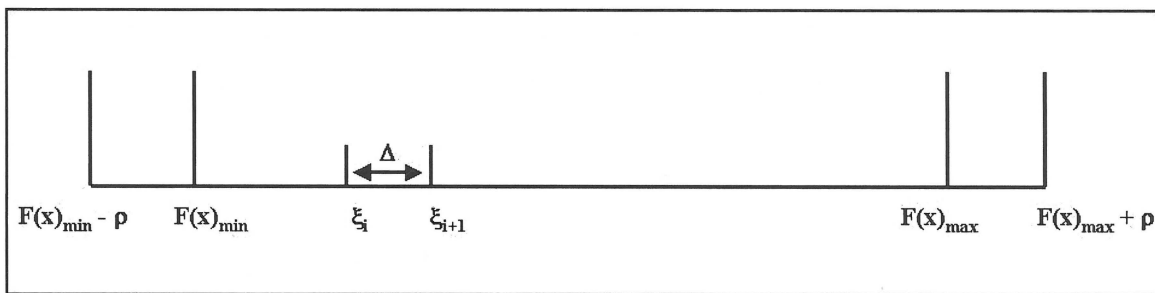
each of those steps. The author's approach interprets Sanner and Slotine's work for one dimension,  $k = 1$ .

For this technique of predicting the next sample, a summation of Gaussian radial basis functions is used to model each datastream. In the case of an image, each pixel is its own datastream. The Gaussian is nonlinear, continuous, and has infinite support.

$g_i(x, \xi_i) = e^{-\pi \sigma_i^2 \ x - \xi_i\ ^2}$	$\ x - \xi_i\ ^2 = (x - \xi_i)(x - \xi_i)$ <p><math>\sigma_i^2</math> is the variance at point <math>i</math> (Gaussian width)</p>
---	--

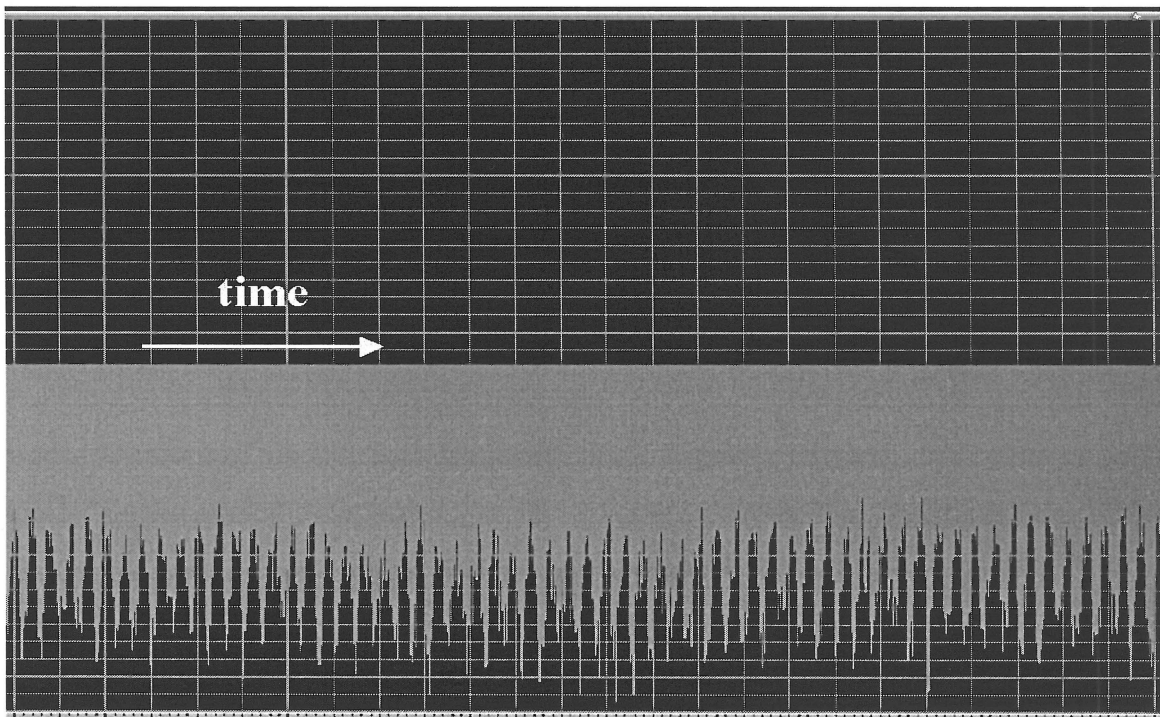
(Note here that the variance,  $\sigma_i^2$ , appears in the numerator of the exponential. This is not the implementation form of the usual Gaussian equation where the variance appears in the denominator. The form of the Gaussian employed here is the result of the original derivation. See Equation 2.3 in Sanner and Slotine, 1991, December.)

Picking the number of Gaussians and their centers,  $\xi_i$ , in the  $k$ -dimensional decision region,  $R^k$ , is a matter of deciding how large the model is to be. (Making the model too large is as ineffective as making it too small.) For  $k = 1$ , the  $\xi_i$  lie spaced evenly along a line segment formed by the range of expected values of  $F(x) \in [F(x)_{\min}, F(x)_{\max}]$  plus a zone  $\rho = \text{some constant}$ .  $\rho$  accounts for errors in estimating the max/min range of  $F(x)$ . The distance between Gaussian centers is  $\Delta \leq 1/(2\beta)$ . These factors are illustrated in Figure 58.

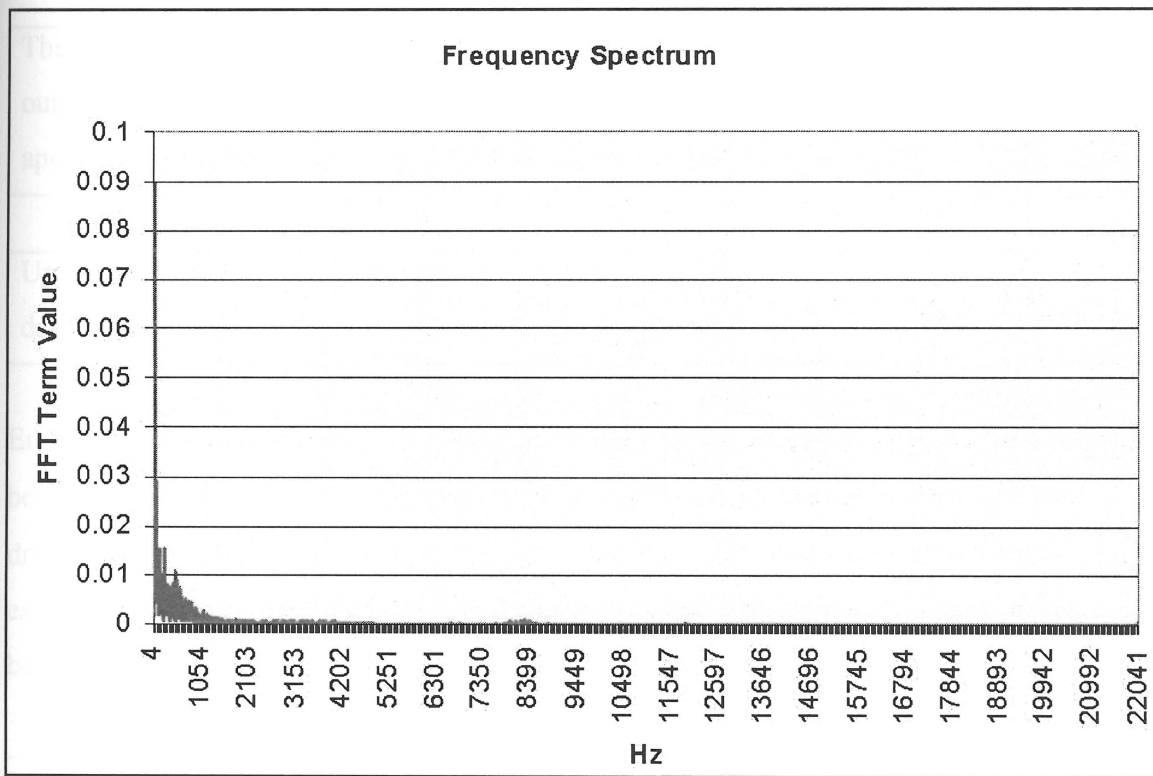


**Figure 58.** Illustration of the Gaussian centers,  $\xi_i$ , and the distance between them,  $\Delta$ . (The max/min range of  $F(x)$  is extended by the constant  $\rho$ .)

$\beta > 0$  is less than the maximum frequency component (in the Fourier Transform sense) used to reproduce  $F(x)$ . Thus,  $\beta$  increases as does the meaningful frequency content of  $F(x)$ . Since the distance between Gaussians,  $\Delta$ , is inversely related to  $\beta$ , the number of Gaussians needed to cover the range in Figure 58 increases as does  $\beta$ . Figure 59 shows an audio signal. Figure 60 is the Fourier Transform plot of that signal. In this case, the maximum meaningful frequency content begins to disappear around 2000hz so that  $\beta \leq 2000$ . The author was able to build a successful predictive model of this audio signal using only 30 terms. However, if it is to be assumed that the signal's control law can change, there is nothing to stop the signal's frequency content from becoming so high or low that the model is no longer useful. This unbounded case is one where it must be recognized that there are implied assumptions in the way the model is constructed. A way around this is to keep a background process running that constantly evaluates the signal's Fourier Transform using a time window. Still, this could lead to the need to modify the number of Gaussians in the model. If this were done, how the heights of the original Gaussians would be distributed in the new model is an open question. At the very least, there would have to be some period of adjustment to establish the new model.



**Figure 59. Original audio signal**



**Figure 60. Fourier Transform plot of original audio signal**

In brief, the number of Gaussians needed to model  $F(x)$  increases as does the meaningful frequency content of  $F(x)$ . Judgment has to be used here. One does not want to attempt to model the high-frequency “noise”. Just capture the meaningful parts of the signal. On the other hand, if the scene is truly very robust, a sufficiently large number of Gaussians is needed to provide an accurate prediction. This author has found that choosing the number of Gaussians to use is not a delicate affair, assuming there are some bounds on how much the signal can change its frequency content.

For applications of low dimensionality such that  $k$  is small (our case is univariate so  $k = 1$ ), a good rule of thumb uses  $\sigma_i^2 = \pi\beta^2$ , making the variance a constant for all Gaussians. Another good rule of thumb has  $\Delta^2 = 1/(8\sigma^2)$ . The best approach is to determine the signal’s spectral components via the Fourier Transform or some other method. The variance follows by calculation. This leads directly to  $\Delta$ , the distance between Gaussian centers, and thereby to the number of Gaussians to use to cover the range illustrated in Figure 58.

The final issue is how to initialize and evolve the output weights. Let $f(x)_t$ be the function approximation due to sample $x$ received at time $t$ .	$f(x)_t = \sum_{i=1}^n [c_{i[t-1]} g_i(x_t, \xi_i)]$
---	--

Updating the output weight, $c_{i[t-1]}$ , due to $f(x)_t$ yields $c_{it}$ .	$c_{it} = c_{i[t-1]} - K_t \varepsilon_{at} g_i(x_t, \xi_i)$
--	--

Empirically, the author has found that it is usually best to initialize all  $c_{i[t=0]}$  to a constant between 0 and 1, non-inclusive. Now let  $\varepsilon_t = (f(x)_t - F(x)_t)$ , the feedback signal that drives weight adjustments. This is the prediction error, the difference between the estimated function result and the actual function result. The estimation is performed before the arrival of the actual result.

$\varepsilon_{at} = \varepsilon_t - \Phi \text{sat}(\varepsilon_t / \Phi)$	$\text{sat}(z) = z$ if $ z  \leq 1$ and $\text{sgn}(z)$ otherwise $\text{sgn}(z) = -1$ if $z < 0$ and $+1$ otherwise $\Phi$ is the minimum expected error
--	---

What this formula says is that the network's output weights are not updated if the prediction error is less than the minimum expected error. Intruder's are postulated when the system is in operational mode and the prediction error is sufficiently large.

$K_t = \frac{G}{\sum_{i=1}^n g_i(x_t, \xi_i)^2}$	$K_t$ is the adaptation gain. The theory requires $G < 2$ . Empirically, $G = 0.1$ appears to work well. $K_t$ must be positive.
--	--

## Appendix B

### Mechanization of Statistical Significance Calculations

If you need to apply statistical measures but are not a statistician then the writings of Bruning and Kintz (1968) are very good. This present research needed a measure of statistical significance to measure the reliability of the technology. Their book presents such a measure complete with a computational process and examples. While most texts concentrate on statistical theory, this book concentrates on application. The result is that the reader understands how to apply the given statistical test and how to interpret the results. The authors cover simple tests like mean, standard deviation, standard error of the mean, and t-tests. They go on to cover analysis of variance (two chapters), correlation, nonparametric tests, tests of significance, and indices of relationship.

The statistical significance of the difference between trials can be computed using the t-Test for Difference Between Two Independent Means, Assuming Equal Variances. This calculation results in a statistical judgment as to whether the performance difference between two independent groups is significant. The most common use assumes equal variances (Bruning and Kintz, 1968, pp. 9-12; also see Spiegel, 1961, p 70).

The mechanization chosen is implemented by an Excel spreadsheet. One sample calculation is shown in Figure 61. The two columns on the left of the figure are the sample data. Each column contains values from members of an independent group. For instance, the values could represent test scores. The table to the right of the figure explains how to interpret the calculation of statistical significance. In this case, there is no statistical significance between the scores of the two groups.

107	109	t-Test: Two-Sample Assuming Equal Variances		
96	94			
88	127		<i>Variable 1</i>	<i>Variable 2</i>
131	76	Mean	99.66666667	102
109	115	Variance	192.4242424	195.8461538
84	121	Observations	12	14
79	87	Pooled Variance	194.2777778	
105	92	Hypothesized Mean Difference	0	
108	91	df (Degrees of Freedom)	24	
92	98	t Stat	-0.425532945	
96	104	P(T<=t) one-tail	0.337119908	
101	96	t Critical one-tail	1.710882316	
	110	P(T<=t) two-tail	0.674239816	
	108	t Critical two-tail	2.063898137	

The computed "t value".

The "t value" that is significant at the 0.05 confidence level for the computed degrees of freedom. If the absolute value of the computed "t value" is less than this number, there is no significant difference between the two groups.

Figure 61. Example calculation of statistical significance

## Appendix C

### Derivation of the Equation for Estimating $f$ in Amdahl's Equation

Amdahl (1967, 1988) derived an equation that predicts the amount of speedup to be expected from some number of processors given a certain fraction of code that has to be run sequentially (cited and explained by Zomaya, 1996, p 14).

$$S_N \leq \frac{1}{(f + ((1 - f) / N))}$$

Where:  $S_N$     the predicted execution time divisor for the single-processor case  
 $N$             the number of processors  
 $f$             the fraction of the code that must be executed sequentially  
                  (loops unwound)

There are times when it is necessary to estimate  $f$  when all the other variables are measured by some experiment. This can be an important calculation since it can demonstrate how much code is currently being run sequentially. Having this number can help justify an effort to work on that part of the code so that more of it can take advantage of parallel processing. The following algebraic process converts Amdahl's equation into an estimate of  $f$ .

$$\frac{1}{S_N} = f + \frac{1-f}{N} = \frac{f^2 N}{fN} + \frac{f(1-f)}{fN} = \frac{f^2 N + f(1-f)}{fN} = \frac{f^2 N + f - f^2}{fN} = \frac{fN + 1 - f}{N}$$

$$\frac{N}{S_N} = fN + 1 - f; \frac{N}{S_N} - 1 = fN - f = f(N - 1); f = \frac{\frac{N}{S_N} - 1}{N - 1}$$



## Appendix D

### Support Provided to this Research

- Financial Support Ball Aerospace & Technologies Corp (<http://www.ball.com>) provided support for tuition, books, and fees.
- Advice Forum Gertrude Abramson, a faculty member at Nova Southeastern University, mediates a very helpful forum giving general and specific advice to doctoral students as they move through their courses and dissertation.
- Forge Preprocessor Converts Java source code to Verilog. Xilinx Corp (<http://www.xilinx.com>) provided access to this tool as part of their beta-test program. Two people from Xilinx (<http://www.xilinx.com>) were instrumental to the success of this part of the research. Don Davis and Jay Gould had quite an adventure. The Forge beta test started off with the idea that Xilinx personnel would be dealing only with experts in FPGAs. However, this author convinced them that someone with no experience whatever would be valuable in helping them reach their strategic goal. This goal envisions growing the population of FPGA programmers by enabling traditional software application developers to target that chip using common high-level languages. This group of latent FPGA programmers knows nothing at all about such devices. Don and Jay were very proactive and patient throughout the project.
- Java Fixed-Point Math Library Onno Hommes from jScience (<http://home.rochester.rr.com/ohommes/MathFP/index.html>) provided his MathFP Java fixed-point library and advice on using the various versions. Onno has created an excellent piece of work that is unmatched from this authors' perspective. A long search revealed nothing like it available anywhere. Such libraries will be essential as more and more traditional developers target FPGAs. He is truly ahead of the pack.

- Single-channel data acquisition hardware and software A WinDAQ DI-194 data acquisition starter kit was provided by DATAQ (<http://www.dataq.com>). This kit includes all software and hardware for sampling a (-10v .. +10v) data stream at a maximum of 240 samples per second, saving the samples to disk, converting the resulting file to various formats, and reviewing the waveforms. It also has various Fourier Transforms that can be applied to the data. All software is fully graphical.
- QuickCam digital color camera Loaned by Donald Bertke, Ball Aerospace Technologies Corp (<http://www.ball.com>). This device produces 30 image frames per second (maximum, depending on lighting conditions). Each frame can be sized at 640x480 30-bit pixels (maximum, depending on user settings). Availability is via [http://www.logitech.com/us/cameras/ca28\\_100.html](http://www.logitech.com/us/cameras/ca28_100.html).
- LookingGlass Machine Vision Kit This is a tool produced by Pong Suvan that consists of a basic platform into which user-developed DLL executables can be linked. The author has been involved with Suvan as a beta tester. Suvan is studying machine vision at the College of Computer Science, Northeastern University, Boston, Massachusetts. His is an excellent kit for real-time image capture and processing that he has released for use by researchers. Using a pipeline approach, the experimenter can employ the provided techniques or install unique ones. Availability is through <http://www.ccs.neu.edu/home/psksvp/lg.htm>.
- Two networked quad-processor systems These were loaned by Ball Aerospace & Technologies Corp (<http://www.ball.com>). They make up the nodes on a virtual machine achieved via a common TCP/IP LAN and deployment of the Message-Passing Infrastructure (MPI). Each system is made by Intergraph and has 4 Intel 200mhz CPUs having a common 500mb memory. The operating system is WinNT v4sp4.

- Audio Processing Tool Support for audio processing came from Goldwave, a tool produced by C.S. Craig that enables audio capture, editing, and display (<http://www.goldwave.com>).
- Engineering Spreadsheet DSP Development Corp (<http://www.dadisp.com>) donated a student copy of their DADiSP engineering spreadsheet. This spreadsheet has hundreds of engineering functions that can be applied to sampled data. It is a serious professional tool whose output is compatible with standard spreadsheets.
- Engineering Plot Software Charles Everding, Ball Aerospace & Technologies Corp, donated his personal copy of engineering data analysis software.
- MathCad Peter Telek, Ball Aerospace & Technologies Corp, donated his personal copy of MathCad.

## Appendix E

### Author Biography

Peter Raeth joined Ball Aerospace & Technologies Corp in January 1998 as a Computer Engineer. He currently designs algorithms and heuristics for spectral exploitation. His experience includes 17 years as an officer, computer research scientist and scientific analyst, United States Air Force (out of 21.5 years active military service). This includes 5 years in industrially funded organizations. His past work was in modeling, simulation, and adaptive automation for passive radar signal identification, cockpit decision aids, computer performance analysis, and data sequence analysis. He has also gained experience in blackbox software test planning, signal analysis, pattern recognition, sensor and data fusion, and parallel/distributed processing. He received the BS degree in Electrical Engineering from the University of South Carolina and an MS in Computer Engineering from the Air Force Institute of Technology. He completed an Air Force career in March 1997. Besides staff billets at Headquarters Air Force Systems Command and Air Force Materiel Command, his assignments included duties in Electronic Warfare flight test support (3246<sup>th</sup> Test Wing), radar signal passive identification (Avionics Directorate, Air Force Research Laboratory), and pilot decision aids (Flight Dynamics Directorate, Air Force Research Laboratory). Just after leaving the service, Mr. Raeth worked with Simulation Technologies, Inc performing duties in distributed simulation, distributed processing, and software blackbox testing. He has given numerous seminars and published many articles, papers, and reports on adaptive automation. He is editor and a contributing author of the book, "Expert Systems: A Software Methodology for Modern Applications", published by the IEEE Computer Society in 1990. In 1991 he was awarded a research fellowship in neural networks at the University of Dayton Research Institute. In 2000 he was awarded third-place in the National Aerospace and Electronics Conference best-paper competition. In 2001 he received Ball Corporation's top award, the Award of Excellence. In 2003 he graduated with a PhD in Computer Science from Nova Southeastern University. Mr. Raeth is a member of the electrical engineering honor societies Tau Beta Pi and Eta Kappa Nu, the community service honor society Omicron Delta Kappa, IEEE Computer Society, and the ACM Special Interest Group in Artificial Intelligence. He has twice won the Armed Forces Communications and Electronics Association's gold medal. The Dayton Affiliate Societies Council has twice selected him as one of the Region's Top 20 Technology Leaders. His web site on adaptive automation was selected in 1998 as a Science Site of the Week by the National Academy Press. Mr. Raeth is a lay minister at his church and enjoys trap, skeet, and sporting clays. He is a member of the Ohio Academy of Science's Science Fair Judging Team and is a faculty member of University of Phoenix' Online Campus. His reading and movie pleasure is science fiction. He also enjoys building scenarios for science fiction video games. He can be reached via [peter\\_raeth@juno.com](mailto:peter_raeth@juno.com).

### Reference List

- Aach, T., Kaup, A. (1995, August). Bayesian algorithms for adaptive change detection in image sequences using markov random fields. *Signal Processing and Image Communication*, ISSN: 0923-5965, 7( 2), 147-160.
- Abutaleb, A.S., Elfishawy, A.S., Kesler, S.B. (1991, May). Adaptive algorithms for change detection in image sequence. *Signal Processing*, ISSN: 0165-1684, 23(2), 179-192.
- Agehed, K.I., Eide, A.J., Lindblad, T., Lindsey, C.S. (1998, Apr). Ram-based neural networks for data mining applications. In *Proceedings of the SPIE The International Society For Optical Engineering, Issue 3390*, 430-439.
- Akin, D.L., Sanner, R.M. (1990, April). Neuromorphic pitch attitude regulation of an underwater telerobot. *IEEE Control Systems Magazine*, ISSN: 0272-1708, 10(3), 62 - 68.
- Amdahl, G.M. (1967). *Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities*. Proc AFIPS, v 30, 483-485.
- Amdahl, G.M. (1988). *Limits of Expectation*. Journal of Supercomputer Applications, 2(1), 88-97.
- Anand, S.S., Patrick, A.R., Hughes, J.G., Bell, D.A. (1998). A data mining methodology for cross-sales. *Elsevier Press, Knowledge-Based Systems*, 10, 449-461.
- Anand, S.S., Bell, D.A., Hughes, J.G. (1995, November). The role of domain knowledge in data mining. In *Proceedings of the ACM International Conference on Artificial Intelligence*, 37 - 43.
- Anandan, P., Shafer, S. (1999). *Vision Software Development Kit*. Bellevue, WA, Microsoft Corporation, <http://research.microsoft.com/Vision/>.
- Andraka, R. (1998). A survey of CORDIC algorithms for FPGA-based computers. In *Proceedings: ACM Conference on Field Programmable Gate Arrays*, <http://www.andraka.com/files/crdcsrvy.pdf>.
- Bailey, D.H. (1991, Jun 11). *Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers*. Technical Report # RNR-91-020, NASA Ames Research Center, Moffett Field, CA.

- Bevington, P.R., Robinson, D.K. (1992). *Data Reduction and Error Analysis for the Physical Sciences*. New York, NY: McGraw-Hill.
- Bharat, R.R., Lu, S.C.-Y., Stepp, R.E. (1990, Sep). Knowledge-based equation discovery in engineering domains. In *Proceedings of the IEEE Eighth Int Workshop on Machine Learning*, 630-634.
- Bhatia, D. (1997). Reconfigurable computing. In *Proceedings: IEEE International Conference on VLSI Design: VLSI in Multimedia Applications*, p 356-359.
- Bigus, J.P. (1996). *Data Mining with Neural Networks*. New York, NY: McGraw-Hill.
- Brachman, R., Khabaza, T., Kloesgen, W., Piatetsky-Shapiro, G., Simoudis, E. (1996, Nov). Mining business databases. *Communications of the ACM*, 39(11), 42-48.
- Bruning, J.L., Kintz, B.L. (1968). *Computational Handbook of Statistics*. Glenview, IL: Scott, Foresman and Company.
- Bruzzone, L., Serpico, S.B. (1997, December 1). Detection of changes in remotely-sensed images by the selective use of multi-spectral information. *International Journal of Remote Sensing*, ISSN: 0143-1161, 18(18), 3883 – 3888.
- Bungay, H. (1999, May). *Course Notes in Environmental Systems Engineering*. Troy, NY: Rensselaer Polytechnic Institute, <http://www.eng.rpi.edu/dept/env-energy-eng/WWW/NEURAL/>.
- Chang, C.-Y., Wang, J.T.L., Chang, R.K. (1996, Jun). Scientific data mining: A case study. *Int Journal Of Software Engineering And Knowledge Engineering*, 8(1), 77-96.
- Chantry, J. (1998). Intruder detectors: Myths of our time. *Facilities*, ISSN: 0263-2772, 16(5 and 6), 128 – 132.
- Chapman, M.C., Bollinger, G.A., Sibol, M.S. (1992, Dec). Modeling delay-fired explosion spectra at regional distances. *Bulletin of the Seismological Society of America*, 82(6), 2430-2447.
- Chen, S., Billings, S.A., Cowan, C.F.N., Grant, P.M. (1990). Non-linear systems identification using radial basis functions. *International Journal of Systems Science*, 21( 12), 2513-2539.
- Chen, S., Billings, S.A., Cowan, C.F.N., Grant, P.M. (1990). Practical identification of NARMAX models using radial basis functions. *International Journal of Control*, 52(6), 1327-1350.

- Chen, T., Chen, H. (1993, Oct 25). Approximation capability to functions of several variables, nonlinear functionals and operators by radial basis function neural networks. *International Joint Conference on Neural Networks*, v 2, 1439-1442.
- Craig, C.S. (1998, August). *Goldwave – A Tool for Audio Capture, Edit, and Display*. <http://www.goldwave.com>.
- Cubberly, W.H. (ed) (1988). *Comprehensive Dictionary of Instrumentation and Control*. Research Triangle Park, NC: Instrument Society of America.
- Cylix, I. (1997, Mar). CORDIC (COordinate Rotation DIgital Computer). *Circuit Cellar INK*, <http://www.ezcomm.com/~cylix/Articles/RobNav/sidebar.html>.
- Davies, E.R. (1997). *Machine Vision: Theory, Algorithms, Practicalities*. New York, NY: Academic Press.
- Davis, G.B. (1974). *Management Information Systems: Conceptual Foundations, Structure, and Development*. New York, NY: McGraw-Hill.
- Davis, D. (2003, Jan). Xilinx Corporation, personal communication.
- DeCoste, D. (1997, Aug). Mining multivariate time-series sensor data to discover behavior envelopes. In *Proceedings of the AAAI Int Conf On Knowledge Discovery and Data Mining*, 151-154.
- Dictionary.com (2002). *The Free On-Line Dictionary of Computing*. <http://www.dictionary.com/cgi-bin/dict.pl?term=On-Line%20Analytical%20Processing>.
- Eberart, R.C., Dobbins, R.W. (1990). *Neural Network PC Tools*. New York, NY: Academic Press.
- Ehlers, M., W.Z. Shi (1996, September 20). Determining uncertainties and their propagation in dynamic change detection based on classified remotely-sensed images. *International Journal of Remote Sensing*, ISSN: 0143-1161, 17(14), 2729 – 2742.
- Elfshawy, A.S., Kesler, S.B., Abutaleb, A.S., (1991, May). Adaptive algorithms for change detection in image sequence. *Signal processing*, ISSN: 0165-1684, 23(2), 179 – 192.
- Eklund, N. (1999). CORDIC: Elementary function computation using recursive sequences. In *Proceedings: 11<sup>th</sup> Annual Conference on Technology in Collegiate Mathematics*, <http://archives.math.utk.edu/ICTCM/EP-11/C27/paper.pdf>.
- Embree, P.M., Danieli, D. (1999). *Algorithms for Digital Signal Processing*. Upper Saddle River, NJ: Prentice Hall.

- Embree, P.M. (1995). *C Algorithms for Real-Time DSP*. Paramus, NJ: Prentice Hall.
- Engels, R., Lindner, G., Studer, R. (1997, Aug). A guided tour through the data mining jungle. In *Proceedings of the AAAI Int Conf On Knowledge Discovery And Data Mining*, 163-166.
- Enomoto, N., Kawasumi, H., Ohata, H., Okazaki, A., Sekii, A. (1997, April 15). Detecting intruders using time-series data by projection pattern of silhouette. In *Electrical Engineering in Japan*. New York, NY: John Wiley & Sons.
- Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. (1996). From data mining to knowledge discovery: An overview. *Chapter in Advances in Knowledge Discovery and Data Mining*, Seattle, WA, American Association for Artificial Intelligence, AAAI Press, 1-36.
- Fayyad, U. (1997, Aug). Data mining and knowledge discovery in databases: Implications for scientific databases. In *Proceedings of the IEEE International Conference on Scientific and Statistical Database Management*, p 2-11.
- Fayyad, U.M, Haussler, D., Stolorz, P. (1996, Nov). Mining scientific data. *Communications of the ACM*, 39(11), 51-57.
- Frost and Sullivan Corporate Authors. (1999). *World Remote-Sensing Data and GIS Software Markets*, Frost and Sullivan Report # 5807-22.
- Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2, 183-192.
- Gillies, A.C. (1991). *The Integration of Expert Systems into Mainsream Software*. Hingham, MA: Chapman & Hall (now Kluwer Academic Publishers).
- Girosi, F., Poggio, T. (1989, October). Networks and the best approximation property. Massachusetts Institute of Technology Artificial Intelligence Laboratory, Memo # 1164.
- Goldman, E.B. (1988). *Computerized Trading Strategies; How to Program for the Stock & Futures Market*. New York, NY: John Wiley.
- Gokhale, M., Gomersall, E. (1997). High level compilation for fine grained FPGAs. In *Proceedings: IEEE Symposium on FPGA-Based Custom Computing Machines*, p 165-173.
- Gokhale, M., Stone, J., Frigo, J., Ahrens, C. (2002). Streams-C: Stream-oriented C programming for FPGAs. Los Alamos National Laboratory, <http://rcc.lanl.gov/Tools/Streams-C/index.php>.



- Gong, P., Ledrew, E.F., Miller, J.R. (1992, March 10). Registration-noise reduction in difference images for change detection. *International Journal of Remote Sensing*, ISSN: 0143-1161, 13(4), 773 – 780.
- Goodwin, M. (1994). Neural processing set to boost sensor technology. *Sensor Review*, 14(3), 20 - 22.
- Groth, R. (1997). *Data Mining: A Hands-On Approach for Business Professionals*. Upper Saddle River, NJ: Prentice Hall.
- Hilborn, R.C. (1994). *Chaos and Nonlinear Dynamics*. New York, NY: Oxford University Press.
- Hinke, T.H., Rushing, J., Ranganath, H., Graves, S.J. (1997, Aug). Target-independent mining for scientific data: Capturing transients and trends for phenomena mining. In *Proceedings of the Int AAAI Conf On Knowledge Discovery and Data Mining*, 187-190.
- Holland, S. (1995, Nov). The application of neural technology and other data mining tools to high throughput screening. In *Proceedings of the Society For Biomolecular Screening Annual Conference*, 177-188.
- Holstein, W.J. (1998, Dec, 21). Data-crunching santa. *US News and World Report*.
- Hommes, O. (2001, Feb 9). MathFP: Fixed point integer math. *white paper, documentation, and software; Science Technologies*, <http://home.rochester.rr.com/ohommes/MathFP/index.html>.
- Hornik, K. Stinchcombe, M., White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2, 359-366.
- Hsieh, P., Landgrebe, D.A. (1996). Automated training sample labeling using laboratory spectra. In *Proceedings of the IEEE Int Geoscience and Remote Sensing Symposium (IGARSS)*, 3, 27-31.
- Hussain, D., Hussain, K.M. (1981). *Information Processing Systems for Management*. Homewood, IL: Richard D. Davis, Inc.
- Hutchings, B., Bellows, P., Hawkins, J., Hemmert, S., Nelson, B., Rytting, M. (2002). Java hardware description language. Brigham Young University, <http://www.jhdl.org>.
- Israelsson, H. (1990, Dec). Correlation of waveforms from closely spaced regional events. *Bulletin of the Seismological Society of America*, 80(6-B), 2177-2193.

- Isshiki, T., Dai, W. W.-M. (1996). Bit-serial pipeline synthesis for multi-FPGA systems with C++ design capture. In *Proceedings: IEEE Computer Society Technical Committee on Computer Architecture*, p 38-47.
- Jarvis, P. (1990, Oct). Implementing cordic algorithms. *Dr Dobb's Journal*, p 152-158.
- Jones, J. (1992, Sep 17). CORDIC math, Parts I and II. *white paper*, University of Washington, <http://programmersheaven.com/zone22/cat171/>, (two separate files).
- Kanemaru, K., Kaneta, M., Kanoh, H., Nagai, T. (1993, October). Image processing method for intruder detection around power line towers. *IEICE transactions on information and systems*, ISSN: 0916-8532, 76(10), 1153 – 1161.
- Kapetanios, E., Norrie, M.C. (1997). Data mining and modeling in scientific databases. In *Proceedings of the IEEE Int Conf On Scientific And Statistical Database Management*, 24-29.
- Kennedy, R.L., Lee, Y., Reed, B. vR. C.D., Lippmann, R.P. (1998). *Solving Data Mining Problems Through Pattern Recognition*. Upper Saddle River, NJ: Prentice Hall.
- Kerestecioglu, Feza (1993). *Change Detection and Input Design in Dynamical Systems*. New York, NY: John Wiley & Sons.
- Kersten, M.L., Siebes, A.P.J.M., Holsheimer, M., Kwakkel, F. (1997, Mar). *Research and Business Challenges in Data Mining Technology*. New York, NY: Springer Verlag, ISBN: 3540625690, Conference: Fachtagung: Ulm; Germany, Source: \* Datenbanksysteme in Buro, Technik und Wissenschaft, 1-16.
- Kloos, C.D., Cerny, E. (ed). (1997). *Hardware Description Languages and their Applications*. New York, NY: Chapman and Hall.
- Knaust, H. (1997, Sep 17). How do calculators calculate. *course notes*, Department of Mathematical Sciences, University of Texas at El Paso, <http://www.math.utep.edu/Faculty/helmut/cordic/wcordic.html>.
- Knittel, G. (1996, Apr). A PCI-compatible FPGA-coprocessor for 2d/3d image processing. In *Proceedings: IEEE Computer Society Technical Committee on Computer Architecture*, p 136-147.
- Koonce, D.A., Fang, C.-H., Tsai, S.-C. (1997). A data mining tool for learning from manufacturing systems. *Computers & Industrial Engineering*, 33(1-2), 27-30.
- Lee, W., Stolfo, S. J. (1998, January). Data mining approaches for intrusion detection. In *Proceedings of the Usenix Unix Security Symposium*, 79-94.
- Lemieux, J. (2001, Apr). Fixed-point math in C. *Embedded Systems Programming*, 44-50.

- Letner, C. (1996, Jul). *Loop Splitting Under Windows NT*. Dr Dobb's Journal, 50-54 and 80-85.
- Light, W. (Ed.). (1992). *Advances in Numerical Analysis, Volume II*. Oxford, England: Clarendon Press.
- Lockhart, R.W. (1995, Jun). Getting started in pc-based data acquisition. *Sensors Magazine*.
- Logitech Engineering. (1999, Apr). *QuickCam Pro Datasheet*. Fremont, CA: Logitech Corp, [http://www.logitech.com/us/cameras/ca28\\_100.html](http://www.logitech.com/us/cameras/ca28_100.html).
- Louca, L., Cook, T.A., Johnson, W.H. (1996, Apr). Implementation of IEEE single precision floating point addition and multiplication on FPGAs. In *Proceedings: IEEE Computer Society Technical Committee on Computer Architecture*, p 107-117.
- Lu, H., Setiono, R., Liu, H. (1996, Dec). Effective data mining using neural networks. *IEEE Trans on Knowledge and Data Engineering*, 8(6), 957-961
- Manago, M., Auriol, E. (1998). Application of inductive learning and case-based reasoning for troubleshooting industrial machines. *Article in text: R.S. Michalski, I. Bratko, M. Kubat, (ed), Machine Learning and Data Mining: Methods and Applications, New York, NY: John Wiley, 173-183.*
- Mas, J.-F. (1999, January 10). Monitoring land-cover changes: A comparison of change detection techniques. *International Journal of Remote Sensing, ISSN: 0143-1161, 20(1), 139 – 152.*
- Mason, J.C., Cox, M.G., (ed). (1987). *Algorithms for Approximation*. Oxford, England: Clarendon Press.
- Masters, T. (1993). *Practical Neural Network Recipes*. New York, NY: Academic Press.
- Mattson, T. (1996). *Scientific Computation*. Chapter in: *Parallel and Distributed Computing Handbook*, New York, NY: McGraw-Hill, 981-1002.
- Moorby, P.R., Thomas, D.E. (1998). *The Verilog Hardware Description Language*. New York, NY: Kluwer Academic Publishers.
- Morgan, D. (1994). *Practical DSP Modeling, Techniques, and Programming*. New York, NY: John Wiley & Sons.
- Morris, C. (ed). (1992). *Dictionary of Science and Technology*, London, England: Academic Press.

- Morse, S.H. (1994). *Practical Parallel Computing*. New York, NY: Academic Press.
- Nagel, H.H. (1981). Image sequence analysis: What can we learn from applications. In Huang. (Ed.) *Image Sequence Analysis*. New York, NY: Springer-Verlag, 19-228.
- Numerous Authors (1998, May, 20). *The Message Massing Interface (MPI) Standard*. Message Massing Interface Forum, <http://www-unix.mcs.anl.gov/mpi/>.
- Oldfield, J.V., Dorf, R.C. (1995). *Field-Programmable Gate Arrays*. New York, NY: John Wiley & Sons.
- Otnes, R.K., Enochson, L. (1978). *Applied Time Series Analysis – Basic Techniques*. New York, NY: John Wiley & Sons.
- Pao, Y-H. (1989). *Adaptive Pattern Recognition and Neural Networks*. Reading, MA: Addison-Wesley.
- Parmee, I.C. (ed). (1998). *Adaptive Computing in Design and Manufacture : The Integration of Evolutionary and Adaptive Computing Technologies With Product/System Design and Realization*. New York, NY: Springer Verlag.
- Pass, S. (1997, Apr). Data Mining: The power of integration. In *Proceedings of the Int Conf On The Practical Application Of Knowledge Discovery And Data Mining*, 25-40.
- Patterson, W. (1990). Field programmable gate arrays get enough speed and density for computer applications. In *Proceedings: IEEE Computer Conference*, p 477-480.
- Peck, J.P., Burrows, J. (1990, Feb). On-line condition monitoring of rotating equipment using neural networks. *ISA Transactions*, 33(2), 159-164, *Regional Seismic Arrays and Nuclear Test Ban Verification Symposium*.
- Pilati, M. (2000, Jan). National Air Intelligence Center, Personal Communication.
- Raeth, P.G. (2000, November, 14). *Machine clustering infrastructures*. CMP Publications, ChipCenter, <http://www.chipcenter.com/eexpert/paeth/paeth038.html>.
- Raeth, P.G. (2001, Oct 31). Sequence Learning Experimenters Kit. *ChipCenter, CMP Publications*, <http://www.chipcenter.com/eexpert/paeth/paeth060.html>.
- Raeth, P.G. (2001, Oct 12). Sequence Learning. *ChipCenter, CMP Publications*, <http://www.chipcenter.com/eexpert/paeth/paeth059.html>.

- Raeth, P.G., Bertke, D.A. (2000, Oct). Finding events automatically in continuously-sampled data streams via anomaly detection. In *Proceedings of the IEEE National Aerospace and Electronics Conference (NAECON, third place best paper)*.
- Raeth, P.G., Bostick, R.L., Bertke, D.A. (1999, Nov). Adaptive data mining applied to continuous image streams. In *Proceedings of the IEEE/ASME Annual Conference on Artificial Neural Networks in Engineering (ANNIE, nominated for best paper)*.
- Raeth, P.G. (1999, Jul). Object interception via trajectory prediction. demonstration produced for CS860, course in artificial intelligence, Nova Southeastern University.
- Raeth, P.G., Gustafson, S.C., Little, G.R. (1994). A basis function approach to programming concurrent voting systems to perform selection tasks. *Journal of Mathematical and Computer Modeling*, 20(3), 73-88.
- Raeth, P.G. (1993, Apr). The effect of personal and business standards on sustained research funding. *Pittsburgh, PA: High-Tech Communications, Advanced Technology for Developers*, 2.
- Raeth, P.G., Gustafson, S.C., Little, G.R., Puterbaugh, T.S. (1992, Jan). *Stretch and Hammer Neural Networks for N-Dimensional Data Generalization*. Cameron Station, VA: National Technical Information Service, Report # WL-TR-91-1146.
- Rinker, R., Hammes, J., Najjar, W.A., Bohm, W. Draper, B. (2000). Compiling image processing applications to reconfigurable hardware. In *Proceedings: IEEE International Conference on Application-Specific Systems, Architectures, and Processors*.
- Russ, J.C. (1999). *The Image Processing Handbook*. Boca Raton, FL: CRC Press.
- Sanner, R.M., Slotine, J.-J.E. (1998, June 20). Structurally dynamic wavelet networks for adaptive control of robotic systems. *International Journal of Control*, ISSN: 0020-7179, 70(3), 405 – 421.
- Sanner, R.M., Slotine, J.-J.E. (1995, July). Stable adaptive control of robot manipulators using "neural" networks. Cambridge, MA: MIT Press, *Neural Computation*, ISSN: 0899-7667, 7(4), 753 – 790.
- Sanner, R.M., Slotine, J.-J.E. (1991, December). Stable adaptive control and recursive identification using radial Gaussian networks. In *Proceedings of the IEEE Conference on Decision and Control*.
- Sanner, R.M., Slotine, J.-J.E. (1991, May/November). *Gaussian Networks for Direct Adaptive Control*. Cambridge, MA: MIT Report NSL-910503/911105.

- Schalkoff, R.J. (1989). *Digital Image Processing and Computer Vision*. New York, NY: John Wiley & Sons.
- Scholtysik, K. (2000, Oct, 9). NT-MPICH Project Description. *Aachen University, Center for Scalable Computing*, <http://www.lfbs.rwth-aachen.de/~karsten/projects/nt-mpich/index.html>.
- Sevenich, R. (1998, Jan). *Parallel Processing Using PVM*. *Linux Journal*, v 45.
- Shanks, M.E., Gambill, R. (1969). *Calculus of the Elementary Functions*. New York, NY: Holt, Rinehart, and Winston.
- Sharma, A.K. (1998). *Programmable Logic Handbook*. New York, NY: McGraw-Hill.
- Sharman, R. (1997, Mar). The business benefits of data mining. *London, England: Unicom, Unicom Seminars*, 49-62.
- Shi, W.Z., Ehlers, M. (1996). Determining uncertainties and their propagation in dynamic change detection based on classified remotely-sensed images. *International Journal of Remote Sensing*, 17(14), 2729-2741.
- Simoudis, E., Livezey, B., Kerber, R. (1995). Using RECON for data cleaning. In *Proceedings of the 1st Int Conf on Knowledge Discovery and Data Mining, AAAI Press*, p 282-287.
- Slotine, J.-J.E., Li, W. (1991). *Applied Nonlinear Control*, Englewood Cliffs, NJ: Prentice Hall.
- Smith, J., Peters, B., Curry, J., Gupta, D. (1998). Prototype software model for designing intruder detection systems with simulation. In *Proceedings of the Society of Photo-optical Instrumentation Engineers*.
- Spiegel, M.R. (1961), *Statistics*. New York, NY: McGraw-Hill.
- Sprague, R.H., Carlson, E.D., (1982). *Building Effective Decision Support Systems*, Englewood Cliffs, NJ: Prentice-Hall.
- Stewart, J.R. (1994). *Modeling the Dispersion of Gaseous Pollutants in Air*. Chapter in: *Parallel Processing*, New York, NY: Addison-Wesley, 235-251,
- Suvan, P. (1999). *Robot Vision*. Real-time image processing software, College of Computer Science at Northeastern University, Boston, Massachusetts, <http://www.ccs.neu.edu/home/psksvp/rv2.htm>.
- Tan, S., Hao, J., Vandewalle, J. (1993, Oct 25). Nonlinear systems identification using RBF neural networks. In *Proceedings of the Int Joint Conf on Neural Networks*, v 2, 1833-1836.

- Taylor, H.R. (1997). *Data Acquisition for Sensor Systems*. New York, NY: Chapman & Hall.
- Thierauf, R.J. (1982). *Decision Support Systems for Effective Planning and Control*, Englewood Cliffs, NJ: Prentice-Hall.
- Terry, H. (1999, May). *Glossary on Neural Networks*. Phoenix, AZ: Knowledge Technology,  
[http://www.primenet.com/pcai/New\\_Home\\_Page/glossary/pcai\\_n\\_o\\_glossary.html#Neural\\_Networks](http://www.primenet.com/pcai/New_Home_Page/glossary/pcai_n_o_glossary.html#Neural_Networks).
- Toulson, D.L., Toulson, S.P. (1999, May). *Neural Net Tutorial*. London, England: Intelligent Financial Systems, <http://www.ifsys.co.uk/html/tutorial.html>.
- Toxvaerd, S. (1994, Jul 25). *Parallel Computations in Molecular Dynamics*. Proc Int Workshop in Parallel Scientific Computing, Berlin, Germany: Springer-Verlag, Lecture Notes in Computer Science (Parallel Scientific Computing), 498-504.
- Volder, J.E. (1959). The CORDIC trigonometric computing technique. *IRE Transactions, EC-8*, p 330-334.
- Way, J., Smith, E.A. (1991). The evolution of synthetic aperture radar systems and their progression to the EOS SAR. In *Proceedings of the IEEE Transactions on Geoscience and Remote Sensing*, 29(6), 962-985.
- Weiss, S.M., Indurkha, N. (1998). *Predictive Data Mining*. San Francisco, CA: Morgan Kaufmann.
- Wenban, A., Brown, G. (1996, Apr). *A software development system for FPGA-based data acquisition systems*. In *Proceedings: IEEE Computer Society Technical Committee on Computer Architecture*, p 28-37.
- Wo, D., Forward, K. (1994, Apr). Compiling to the gate level for a reconfigurable co-processor. In *Proceedings IEEE Workshop on FPGAs for Custom Computing Machines*, p 147-154.
- Xilinx Corporate Authors. (2001, Feb 8). *Forge User's Manual - Version 0.5 - Preliminary*. Columbia, MD: Xilinx, Introduction.
- Yalamanchili, S. (1997). *VHDL Starter's Guide*. New York, NY: Prentice Hall.
- Zhong, N., Dong, J., Ohsuga, S. (1998, Jun). *Soft Techniques to Data Mining*. New York, NY: Springer, Lecture Notes In Computer Science, Issue 1424, p 231-238.
- Zomaya, A.Y.H. (1996). *Parallel and Distributed Computing Handbook*. New York, NY: McGraw-Hill.