SIGNAL PROCESSING WITH NODAL NETWORKS ON A SIMD MASSIVELY PARALLEL PROCESSOR

William I. Lundgren

GE Aerospace Advanced Technology Laboratories
ATL Building, Moorestown, NJ 08057

Abstract

The goal of the work reported in this paper is to develop methods for using a large scale SIMD parallel processor such as the Connection Machine for signal processing. The primary focus of the work reported in this paper is to develop nodal network methodologies that can be used to effectively implement algorithms that range from signal processing to discrete logic systems. One benefit of nodal network methodologies is their inherent SIMD nature. As a result their implementations are closely related to the architecture of SIMD parallel processors like the Connection Machine. Also, the homogeneity provided by such systems allows the uniform integration of signal processing and discrete logic systems. As a first step we have implemented two versions of an algorithm to track formants in speech. The first implementation uses data parallel coding techniques. The second implementation uses a nodal network. The algorithm contains logic that, at every frequency/time point in a spectrogram, chooses between several filters to find the filter that best matches linear energy structure at that point. The choice of filter at each point is determined on the basis of information in adjacent points. The nodal network implementation of the algorithm uses only two node types, a fuzzy AND and a fuzzy OR (henceforth referred to as AND nodes and OR nodes respectively). The connections between nodes can be either non-inverting or inverting. The inverting effectively produces a NOT. The algorithm relies on the parameters associated with each node and connectivity between the nodes to simulate the original algorithm. The result is a nodal network "programmed" to identify formants in a spectrogram. The two implementations are comparable in performance and speed of execution. The conclusion is to continue the investigation of this type of nodal network.

1.0 Introduction

The long term goal of this work is to develop nodal network techniques for large vocabulary speaker independent speech recognition. In speech recognition, the biggest problems are that small changes in the acoustic signal can change the word recognized, and conversely, acoustically different speech signals may be recognized as the same word. Systems that use data reduction to decrease the size of the input vector smooth the signal and remove some distinctive features of the speech. Large input vectors are necessary if the input is to contain enough information to make fine distinctions between acoustic inputs. Further, the classifier must be able to selectively focus on small or large amounts of acoustic information as required. This provides the capability to build a classifier that is a complex mapping between the acoustic input and the words recognized. Unfortunately these two characteristics, a large input vector and flexible use of the acoustic input, make the task of training very difficult.

Nodal networks provide the capacity and flexibility required. The network described below was designed to provide a means for dealing with large input vectors, complex classifiers and, eventually, automatic training. A high level diagram of the sought after system with training is shown in Figure 1. The system is divided into

![Figure 1. High Level Diagram of Future System.](image)

CH2649-2/89/0000/0663$01.00 © 1988 IEEE
a classifier and a trainer. The classifier consists of a large network of nodes, an input preprocessor and a post processor. The automatic training consists of a node configuration module and a knowledge base. Note that rather than applying the knowledge during the classification process, the knowledge is used off line to train the network. The heart of the problem is to decide how the network configuration module should choose a network configuration to address a particular recognition task. For even the very simple network topology described below using a modest set of nodes (10 levels, 1000 nodes at each level) leads to a huge set of potential solutions. It is easy to establish a lower bound of 10 to the power of 60,000. The system described below does three things to help the node configuration module select a meaningful solution. It uses:

1. a knowledge base to guide the training of the network from the top down.
2. well established node configurations that can be used for specific types of tasks.
3. consistencies in the acoustic input to drive the training from the bottom up.

The knowledge makes it possible to train the network in spite of a small amount of training data compared to the size of the solution space.

This paper focuses on the development of nodal network configurations for processing data. After the configuration of nodes for various tasks are adequately understood, the effort will be shifted to the automatic training of the nodal networks. The approach to training proposed in this paper differs from that proposed by others (Refs. 1, 2). Some basic ideas for the training of the nodal network are presented in the appendix of a technical report by Dave Graff and the author (Ref 3). The methodology discussed there is manual training. That methodology was developed with the intent of eventually automating the procedure and introducing a knowledge base to produce an automatic training capability.

2.0 The Nodal Network

The nodal network is designed to provide the following two features:

1. Hierarchical processing of the data. The goal is to provide layers of information, each slightly more abstracted from the data. The hierarchical nature of the system provides information at each level on which future training can be based. Hence, during the course of training, an instance of the nodal networks, one would have an increasingly complete set of fundamental nodes. As the set of nodes becomes more complete the training will become easier.

2. Uniform integration of "symbolic" and numerical processing. Each node in the system can be thought of as a numerical value and symbol pair. For example, the input nodes described below are a Power Spectral Density (PSD) of the speech signal. The node can be labeled "Spectral Power at Δf * n hz." Δf is the frequency resolution of the PSD and n is the number of the PSD nodes. Higher in the network the node values approximate a likelihood. For example, one of the nodes described below will be "band of spectral power at Δf * n hz and angle k".

The intent is to develop a knowledge base and a set of nodes that are directly related. This will ensure that the node configuration module will be able to use the knowledge base to assist in the configuration of the nodal network. Further, the hierarchical nature of the network will provide knowledge that is increasingly abstracted from the acoustic data and more directly related to the words. The homogeneous representation of knowledge at every level of the system makes feedback between any levels in the system possible. It also provides for homogeneous access to all levels of information when making a decision.

2.2 Description of the Nodal Network

A nodal network can be fully described by specifying the node layout, the interconnectivity of the nodes and the definition of how the nodes process their inputs. An example of the node layout and interconnectivity is shown in Figure 2. There are multiple levels of nodes (10 or 20 levels is reasonable) and in the current system there are two inputs to each node. Past implementations by the author have used four or five inputs to each node. Any network using two or more inputs can be implemented with a networks using only two inputs to each node. At this point it is not clear that there is any advantage to using a larger number of inputs per node. Feedback is a potentially powerful device not used in the current network. Figure 2 shows one example of feedback. The role of feedback and stability of networks with feedback will be the subject of future work.

The nodes are simple in behavior and similar to some of the other nodes used in the neural network community (Ref s.1, 2). The inputs are each transformed with a linear equation. The slope and the intercept used are unique for each input of each node in the network. The result is then thresholded to maintain the value Two inputs to each node

Figure 2. Node Layout and Interconnectivity.
between 1.0 and 0.0. The inputs can be “inverted”, i.e. after transformation and thresholding the result can be subtracted from 1.0 to produce a NOT of the input. The two inputs to a node are either summed and divided by 2 (to insure the input remains between 1.0 and 0.0) or they are multiplied. The summation produces a node that behaves like an ORing of the inputs. The multiplication produces a node that behaves like an ANDing of the inputs. Two examples of nodes are shown in Figure 3.

\[ y_1 = \text{th}(2.5 \cdot x_1 - 1.0) \]
\[ y_2 = \text{th}(2.5 \cdot x_2 - 1.0) \]
\[ y = y_1 \cdot y_2 \]

Figure 3a. AND Node.

\[ y_1 = \text{th}(2.5 \cdot x_1 - 1.0) \]
\[ y_2 = 1.0 - \text{th}(2.5 \cdot x_2 - 1.0) \]
\[ y = (y_1 + y_2) / 2.0 \]

Figure 3b. OR Node.

3.0 Approach

The approach is centered on the use of a relatively simple application as a tool to explore programming of nodal networks. The application chosen for this study was recognition of formant peaks in a time-frequency-power representation of a speech signal. Two implementations of an algorithm to identify formant peaks were coded for comparison. The first used standard coding techniques with one processor assigned to each element in the time-frequency plane. The second implementation used a similar processing string, but was coded with a network of fuzzy AND and OR nodes. Each node in the network was assigned to a processor.

This approach was chosen because it illustrates:
1) The feasibility of using a nodal network to perform a signal processing task and 2) the design of a basic nodal network to perform some functions required by a signal processing task.

4.0 The Application

4.1 The Input Data

Formants are bands of energy in the spectral representation of speech. They are a result of the shape of the vocal tract and the associated resonances at the time the sound was produced. For this study, the time waveform of speech data was passed through an anti-aliasing filter with a cutoff at 4000 hz and was then sampled at a 10 kHz sampling rate. The analysis frames were hand aligned to the pitch periods in the time waveform (see Figure 4). This removed much of the temporal variation in the magnitude of the formants. (One of the next tasks will be to develop a spectral analysis network that automatically centers analysis frames on the pitch periods.) The analysis windows were hamming windows 128 samples long. The FFT produced a spectrum of 64 values ranging from DC to 4922 hz in 78.1 hz increments. An example of the formants used in this study are shown in Figure 5 (next page). The spectrogram shown is of an adult male speaker. The pitch period for this speaker averages about 10 msec.

The formants extend for a finite duration in the time direction and move up or down in frequency over time. The objective of the algorithms coded for this effort is to track those formants as they start, stop and move through time.

Figure 4. Segement of Speech Data. Each repetition of similar data is a pitch period.
4.2 Processing String - Conventional Algorithm

1) The first step is to convolve the image with 19 elongated kernels oriented at angles between -45 degrees and +45 degrees (0 degrees is aligned with the time axis). The idea of using elongated filters at different angles was first suggested by Dr. John Meckley (Ref 4). At each time-frequency point the kernel most closely aligned to the energy structure in that region is selected. The kernel is selected by choosing the convolution result that changes the most as the kernel is shifted along the frequency axis. If two or more results are the same size the angle nearest to 0 degrees is chosen. The value of that convolution becomes the new value at that pixel. The angle of orientation of the largest convolution result is also recorded. The result of this matched filtering is an enhancement of the formant peak energy relative to the other energy in the signal.

2) The new image is then rescaled at each point by thresholding with the average energy in nearby locations.

3) The data is then spread along the angle chosen in step one: a) The data is multiplied by 1/3 and added to the data samples which are two data points away along the chosen angle (both backward and forward along the angle). b) The data is then multiplied by 2/3 and added to the data in the adjoining points along the chosen angle.

This process, spreading the likelihood of being a peak along the angles selected in step one, would be equivalent to low pass filtering if all angles were aligned.

4) The final step is to again threshold the result of step four to produce a binary output image.

The implementation on the Connection Machine was accomplished by assigning one data point to each processor and simultaneously working on 128 spectrums of 64 data points each. The rescaled data and the final results are shown in Figure 6.

4.3 Processing String - Nodal Network Algorithm

The processing steps used in the nodal network are similar to those used in the conventional algorithm. The primary difference is a change in their sequencing.

1) The first step is low pass filtering (LPF) of the data along the time axis. This smooths out some of the irregularity in the spectrum that results from noise added to the formant structure.

2) A local average LPF spectrum is calculated based on 8 data points. Again this is along the frequency axis. This result is maintained independently of the LPF spectrum from step 1.

3) The LPF spectrum is then set to zero if it is less than the average in its area, to one if it is 1/8 or more bigger and otherwise scaled linearly between those two values. This rescales the whole spectrum relative to the average in the area around it.

4) The data is then reduced further by determining if each point is a peak in the modified energy along a given time slice.

5) The final step is to determine if there is a continuity of peaks along each of the possible angles.

The results at Steps 3, 4 and 5 are shown in Figure 7. For completeness, all node definitions are given in Table 1. The first three layers of the nodal network are diagrammed in Figure 8.

Figure 6a. Conventional Implementation - Rescaled Data.

Figure 6b. Conventional Implementation - Finale Result.
Explanation of table:
Each node is repeated for every element (64 of them) in the spectrum. The first array index is the offset along the frequency axis. The second index is the offset along the time axis. Note that all time offsets are less than or equal to zero. This is because the "recent" history of a node is kept but future values are unavailable. If the time offset is 0 the second index is omitted. The name to the left of the equal sign is the name of the node. The two (or one) names to the right of the equal sign are the names of the input nodes. If a pair of numbers enclosed in braces follows the node name that input is transformed using the first number as the slope and the second as the intercept. If a pair of numbers is not present the slope is by default 1.0 and the intercept 0.0. If a NOT follows the node name (or the braces) the input is inverted.

New input data is inserted every time iteration.

Low pass filter (LPF) the input along the frequency axis:

<table>
<thead>
<tr>
<th>Node</th>
<th>Input1</th>
<th>Input2</th>
<th>Slope</th>
<th>Intercept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpf_{1}[0]</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alpf_{2}[0]</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alpf_{3}[0]</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reshape the data to remove low amplitude energy:

<table>
<thead>
<tr>
<th>Node</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>scl_{1}[0]</td>
<td>+</td>
</tr>
<tr>
<td>scl_{2}[0]</td>
<td>+</td>
</tr>
</tbody>
</table>

Identify local maximum along the frequency version:

<table>
<thead>
<tr>
<th>Node</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>pk_{1}[0]</td>
<td>+</td>
</tr>
<tr>
<td>pk_{2}[0]</td>
<td>+</td>
</tr>
</tbody>
</table>

Identify local maximum along the time version:

<table>
<thead>
<tr>
<th>Node</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_{01}[0]</td>
<td>+</td>
</tr>
<tr>
<td>C_{01}[0]</td>
<td>+</td>
</tr>
<tr>
<td>C_{00}[0]</td>
<td>+</td>
</tr>
</tbody>
</table>

Table 1. A Listing of All the Nodes

<table>
<thead>
<tr>
<th>Node</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_{00}[0]</td>
<td>+</td>
</tr>
<tr>
<td>C_{01}[0]</td>
<td>+</td>
</tr>
<tr>
<td>C_{02}[0]</td>
<td>+</td>
</tr>
</tbody>
</table>

Determine if one of the continuities of length 4 is large. Input threshold (8, -5.) requires three out of the four input peaks be large.

There are actually only 15 continuities. The input to the first node is multiplied by 4 (rather than the 8 used in the other nodes) so that its contribution will be the same as the other 14 nodes. The final scaling requires that 1 of 16 continuities be large.

Identify Continuities along time version of length 2:

<table>
<thead>
<tr>
<th>Node</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_{00}[0]</td>
<td>+</td>
</tr>
<tr>
<td>C_{01}[0]</td>
<td>+</td>
</tr>
<tr>
<td>C_{02}[0]</td>
<td>+</td>
</tr>
</tbody>
</table>

Figure 7a. Nodal Implementation - Rescaled Data.

Figure 7b. Nodal Implementation - Formants Identified.

Figure 7c. Nodal Implementation - Continuity at -45°

Figure 7d. Nodal Implementation - Final Result
The nodes shown in Figure 8a are low pass filter (LPF) the inputs. The connections shown by the heavy lines indicate the paths that lead to a single output node. Notice that the input nodes contribute through multiple paths to the output nodes. The numbers at the bottom indicate the relative contribution of each of the nodes. Notice that the scaling of inputs is consistent because at every level all values are divided by 2.

The nodes in Figure 8b calculate the average of the LPF. The three levels together produce outputs that are the sum of 8 LPF nodes. Notice that the input nodes contribute only through a single path to the output nodes. The division by two at each of the three levels accumulates to a division by 8. The result is an exact average of the 8 input connected by the heavy lines.

The nodes in Figure 8c perform a rescaling by summing the LPF value (lpf) and the inverted average (Alpf). The intermediate result (scl1) is:

\[ scl1 = \frac{\text{lpf} + (1 - \text{Alpf})}{2} \]  

or reducing

\[ scl1 = \frac{1}{2} + \frac{\text{lpf}}{2} - \frac{\text{Alpf}}{2} \]  

The next node multiplies this result by 8 and then subtracts 4 to give the output z. Inserting and reducing gives:

\[ z = 4 \times (\text{lpf} - \text{Alpf}) \]  

Notice that the result is 0 if Alpf is greater than lpf. It is equal to one if lpf is greater than Alpf by more than 1/4.

The implementation of this algorithm on the Connection Machine is accomplished by assigning one processor to each data point in the network. The spectrums, each with 64 data points, are pipelined through the system. In fact, the system is being used in a "compute level" parallelism fashion.

5.0 Results

The figures in the previous section illustrate that the algorithms do work for the example given. Note that the single spectrogram evaluation is very insufficient but, the goal of this effort was to illustrate the use of nodal networks not to produce a "fieldable" algorithm. Both algorithms perform similarly on this spectrogram.

The speed of execution of the two implementations indicates that there is not a strong difference in the efficiency of the two algorithms. The ratio of execution time is approximately 4:1 (nodal network to conventional). It must be pointed out that performance was not a focus of this work. The conventional implementation was mildly massaged to improve the performance. No such massaging was done for the nodal network algorithm. Further, the nodal network algorithm was implemented in floating point and the conventional algorithm in 8-bit integer. The nodal
network implementation can be easily implemented in fixed point and the data communications can be dramatically improved without much effort. It is very likely that the execution time of the two systems would then be comparable.

6.0 Conclusions
The first conclusion suggest the continuation of this work. Nodal networks can provide a method for parallel processing one dimensional signal data in a massively parallel SIMD processor if the processor has a reasonably efficient system for random interprocessor communications. There are certainly a lot of questions left unanswered about the long term system. Still the experience is encouraging. The software that supports "programming" of the nodal network is improving and, in turn, making the exploration of these systems much easier.

7.0 Acknowledgements
I want to thank my wife Amy and my children Jeremy and Autumn for their support during the work leading up to this paper and during its preparation.

8.0 References