A SIGNAL PROCESSING ARCHITECTURE BASED ON RAM TECHNOLOGY

Jim Torresen
Vidar Engh Skaugen
Department of Informatics
University of Oslo
P.O. Box 1080 Blindern
N-0316 Oslo, Norway
E-mail: {jimtoer,vidarsk}@ifi.uio.no

KEYWORDS
Electronics, Parallel methods, Performance analysis, Signal processing, Signal processors.

ABSTRACT

A signal processing hardware based on a look-up table (LUT) is presented in this paper. This provides a fast response as well as being compact in size. The paper contains a set of new algorithms to program the look-up table. Experiments have been undertaken for prosthetic hand control. The results are promising.

INTRODUCTION

This paper considers an architecture for signal processing to be applied to complex real-world applications. There are many roads into designing signal processing architectures. These include statistical analysis, neural networks and evolutionary computation. However, in this paper a *look-up table* is investigated for signal classification. That is, having the inputs connected to the address lines of a random access memory (RAM) device with the response given on the data bus output – see Figure 1. The challenging task would be to design the algorithm to fill the RAM during training to obtain the best possible generalisation.

Research has been undertaken using RAM as a neural node (Aleksander and Morton, 1991). By having a set of such nodes, weightless neural networks are implemented. One of the benefits of this approach is the easy implementation with conventional computer hardware.

The idea of the work to be presented in this paper is to apply RAM technology for applications with a *limited* number of inputs. In this way, we are able to use a *single* RAM device addressing all possible input vectors. The requirement would be that the applications could work well with a limited signal resolution. This is to avoid excessively large tables. For example, 16 input bits correspond to a table with 2^{16} = 65536 entries (64K). Doubling the number of input bits to 32 requires a table with 2^{32} entries, or 4 Giga entries.

One appropriate application for this approach is prosthetic hand control – operated by electromyography (EMG) signals (Scott and Parker, 1988). The signals from the remaining part of the arm are of limited resolution and experiments have shown that four bits coding is sufficient for each EMG channel. Four channels are used and thus, the input vector consists of 16 inputs. Implementing this in a controller would be no problem, as even the smallest commercially available RAM chips today far exceed 64K in capacity. Six different motions should be distinguished: Open and Close hand, Extension and Flection of wrist, Pronation and Supination of wrist. The interfaces to the controller are illustrated in Figure 1. The data set consists of the same motions as used in earlier work (Kajitani et al., 1999), and it is collected by Dr. Kajitani at Electrotechnical Laboratory in Japan. The data set consists of 300 vectors and this is only 0.5% of the complete RAM. Thus, to provide generalisation we should fill the other locations according to what is the *closest* motion. This paper contains proposals of a set of novel algorithms for how this can be undertaken.

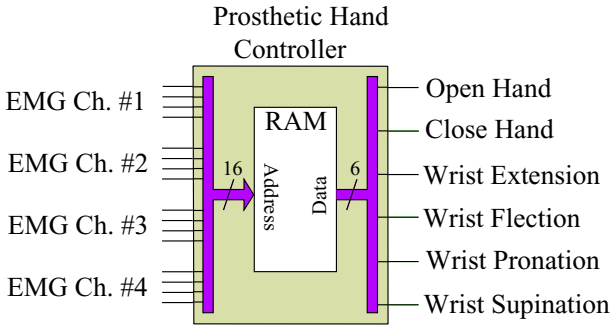

Figure 1: Illustration of the Controller Interfaces

The main benefit of such an approach compared to a neural network based controller (Fuji, 1998) is that it provides a more compact hardware implementation (smaller hardware and less computation). Controllers have also been designed using evolvable hardware (Kajitani et al., 1999; Torresen, 2001), but the evolution time is difficult to predict compared to the approach presented in this paper. For the given data set used in the following experiments, the time used for programming the RAM was approximately equal to the time used for training a feed-forward neural network – solving the same problem in the best possible way. However, the main advantage of this scheme is the fast *runtime* speed, which is constant and given by the read cycle time for the RAM device (typically a few ns). This would be several orders of magnitude faster than neural network computation.

LOOK-UP TABLE ALGORITHMS

In this section, various algorithms for defining the content of the look-up table are introduced.

The data set consists of 50 inputs for each motion, with six possible motions bringing the total number of inputs to 300. Thus, only *one* prosthetic motion should be active at a given time on the output of the controller. Further, two such data sets are available: One for training (programming the RAM) and one for testing (validating performance after RAM programming). The published results on adaptive controllers are usually based on data for non-disabled persons. Since it is possible to observe the hand motions, a good training set can be generated. For the disabled person this is not possible since there is no hand to observe. The person would have to by himself distinguish the different motions. Thus, it would be a harder task to get a high performance for such a training set. This resulted in input signals for different motions being overlapped. That is, the training data contained several occurrences of the *same* input bit pattern, and these occurrences often mapped to *different* motions.

To discover the extent of the problem, we decided to investigate how many wrong predictions there were with an algorithm which had been perfectly trained. On data which did not overlap at all, a perfectly trained algorithm would have a 100% correct prediction rate if given the same input as it was trained with. A table with 64K entries was created. Six values were assigned in each entry to count the *number* of times that a particular input bit pattern was mapped to each of the six possible motions on the output. This table was used for all the experiments presented in this paper. When using the training data both for training and testing the algorithm, there would always be a mapping from the input vector to an output motion. All one needed to do was to look up the input bit pattern in the table and choose the motion with the highest number of mappings. After running through the training data, 270 of 300 inputs (90%) were mapped correctly. In other words, with a perfectly trained algorithm, using only the same data it was trained on, the maximum success rate was already at 90%. Using the same procedure on the test data resulted in 261 of 300 correct predictions (87%).

Then the test set was applied on the table filled by the training data as described above. Only 55 values were predicted correctly. This is not surprising, since many of the input bit patterns in the test data were not mapped to an output motion at all. To get an approximation of the maximum attainable success rate for the test set, we then created two tables, one from the training data and one from the test data. The test data was then used by an algorithm that first checked the table created from the training data, and predicted the best output motion based on the input bit pattern. If the input bit pattern did not exist in the table, it used the table created from the test data instead. This procedure emulated an algorithm – trained perfectly on the training data, which is able to extrapolate perfectly to data not included in the training data. The result here was 209 correct predictions – a success rate of 69.7%. Thus, this is the maximum obtainable test set performance for an algorithm which has been trained perfectly on the training set.

Now the limits of the data sets have been determined and next various algorithms are outlined where only the training set is used during the programming of the table content. The proposed algorithms include a variety of ways to fill the empty locations in the RAM after the training set has been stored. Finally, the test set is applied to compute the generalisation performance.

Smallest Distance

This algorithm finds the input vector(s) from the training data with the least number of bits different from the input being tested. For example, if the *closest* input in the training data has 2 bits different, all inputs in the training data with 2 bits different are selected, and the motion with the highest *count* among them is assigned to be the correct output. This is illustrated by a simplified example below:

Input	# of bits different	# motion count $M_5 - M_0$ (RAM)
100	0	000000
000	1	000000
101	1	000000
110	1	000000
001	2	000001
010	2	004000
111	2	000000
011	3	000600

We are here looking for the smallest distance from the input “100”. The table has sorted other possible inputs according to number of bits different from the one being tested. No match is found for 1 bit difference. For two bits difference, there are two matches found: “001” (motion M_0) and “010” (motion M_3). Since motion M_3 has the highest count, “010” represents the “smallest distance” from “100”. Thus, the content of the RAM for input “100” becomes updated to “001000”.

Results:

- Training data: 270 out of 300 correct.
- Testing data: 151 out of 300 correct.

First Unique

This algorithm is equal to “Smallest Distance”, except that if several of the highest motion counts are equal, the inputs one step further away are used instead, and so on.

Results:

- Training data: 270 out of 300 correct.
- Testing data: 152 out of 300 correct.

Average

All training input vectors with N or less bits *different* from the input being tested are selected and the counts for each motion are summed. The motion with the highest total count among these is assigned to be the correct output. If no training vectors have N or less bits different, the "Smallest Distance" algorithm is used instead.

Results:

N	000	001	002	003	004	005	006	007
Training	270	222	195	185	162	151	128	121
Testing	151	157	162	169	160	157	156	144

First N Vectors

Select all input vectors with 0 bit different from the input being tested. If there are at least N vectors that map to a *single* motion, the motion with the *highest* total count is assigned to be the correct output. Otherwise, add all input vectors with 1 bit different from the input being tested to the selected set and so on until at least N input vectors that map to a single output motion have been found.

Results:

N	001	002	003	004	005	006	007
Training	270	259	245	237	223	198	197
Testing	151	150	154	152	152	166	161

N	008	009	010	011	012	013	014
Training	187	187	179	172	172	173	173
Testing	156	160	165	169	169	172	171

N	015	016	017
Training	170	166	164
Testing	170	168	167

First N Unique

Similar to "First Unique". Once the first unique input vector is found, add one point to the corresponding motion. If that motion now has N points, assign it to be the correct output, otherwise continue with the inputs one step further away. If N unique vectors cannot be found, use the motion with the most points instead.

Results:

N	001	002	003	004	005	006	007
Training	270	230	217	183	186	186	186
Testing	152	174	173	168	166	166	166

DISCUSSION

The algorithms based on the N parameter showed the best test set performance. The maximum was obtained by "First N Unique" for N = 2 with 58.0% correct recognition. This is slightly below the performance using neural networks which achieved 58.8% performance for the same data set (Torresen, 2001). However, the response would be much faster and less computing power is required. This makes the look-up table approach very competetive.

CONCLUSIONS

A signal classification system based on RAM as a look-up table has been presented in this paper. Several algorithms for programming it have been proposed. The results are promising for a prosthetic hand control data set.

REFERENCES

Aleksander, I. and Morton, H. (1991). *An Introduction to neural computing*. Chapman and Hall.

Fuji, S. (1998). Development of prosthetic hand using adaptable control method for human characteristics. In *Proc. of Fifth International Conference on Intelligent Autonomous Systems*, pages 360–367.

Kajitani, I., Hoshino, T., Kajihara, N., Iwata, M., and Higuchi, T. (1999). An evolvable hardware chip and its application as a multi-function prosthetic hand controller. In *Proc. of 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 182–187.

Scott, R. and Parker, P. (1988). Myoelectric prostheses: State of the art. *Journal of Medical Engineering and Technology*, 12(4):143–151.

Torresen, J. (2001). Two-step incremental evolution of a digital logic gate based prosthetic hand controller. In *Evolvable Systems: From Biology to Hardware. Fourth International Conference, (ICES'01)*, volume 2210 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag.