Chapter 3

Dynamic Time Warping (DTW) based Speech Recognizer
3.1 Connected Word Recognizer on a Multiprocessor System

In this chapter, we discuss a speech recognizer called Connected Word Recognizer (CWR) based on dynamic time warping technology. It belongs to the 1st generation of speech recognizers. CWR on a Multiprocessor System addresses the challenges in the area of

- algorithms,
- real time processing with minimal resources and
- application design of a Finite State Automaton to improve the recognition accuracy.

This chapter addresses the issues involved in partitioning and allocating tasks in a multiple-processor environment to maximize throughput, and discusses the implementation of a grammar-driven Speaker-dependent connected word recognizer (CWR).

Many speech recognition algorithms extract a feature vector from the input signal at a rate of 25 to 50 times per second. Vocabulary words may then be represented by sequences of feature vectors each representing the spectral content of the signal over a short period of time called a frame. In the recognition process, new feature vectors are computed at the frame rate (25 to 50 Hz) and compared to every reference vector in every vocabulary word. Comparison involves Euclidean distances between N-element vectors, where N is typically between 10 to 20, and dynamic programming to optimally time-align reference vectors with the input speech vector. This process is computationally demanding and limits the size of the active vocabulary that can be processed in real-time. One way to overcome this limitation is to use multiple programmable processors to distribute this loading. Texas Instruments has developed a multiple processor architecture called Odyssey. In a research environment the multi-processor programmability is extremely desirable since such architecture can be used as a prototype to test and...
evaluate advanced robust speech recognition/DSP algorithm.

The Odyssey system is an expandable, multiple digital signal processor (OSP) architecture based on the TMS32020 programmable microcomputer. Key features of the board are: 20 million multiply/accumulates per second, 512K bytes of data space, and expandability to 16 boards on a NuBus host.

The Odyssey host is Texas Instruments' Explorer, a LISP machine workstation. Software has been provided which extend the high productivity environment of the Explorer into the area of digital signal processing. This provides an environment to perform many intelligent signal processing tasks by associating meaningful relationships between quantitative (signal processing) and qualitative (symbolic processing) entities to develop inferences using expert system technology. Applications such as grammar-driven connected-speech recognition, neural network simulation, and generation or speech with natural language generation techniques are some of the tasks that can utilize the computational power of the multiple DSP and symbolic processing.

3.2 Design considerations

The design and architecture of this multiple processing system was influenced by several considerations some of which are listed below.

- The digital signal processing modules must be readily programmable for maximum algorithm flexibility.
- Each DSP module must be autonomous to allow true parallel execution at maximum speeds.
- Communication between the Explorer LISP host and the DSP modules must be efficient from both the host and the DSP points of view to provide high interaction between Odyssey and Explorer.
- Inter-module communication must be simple and independent of DSP algorithms to allow for easy trans-
port of algorithm modules from other architectures, and yet efficient to maximize data transfer.

- Processing power must be extensible, first by incorporating the largest number of DSP modules per board, then extending to additional Odyssey boards with no unique protocol requirements across boards.
- The Odyssey board must have its own input/output capability to allow for real-time data collection and output.
- The DSP architecture must be flexible enough to accommodate upgrades to the processor and memory without significant impact on software (assuming compatible next-generation DSP chips. Also, the architecture must lend itself to insertion in new hosts with minimum impact on DSP software.

### 3.3 General Architecture description of the multiprocessor system

The Odyssey architecture is designed to provide maximum autonomy to individual digital signal processing devices, while also allowing the DSPs to communicate with each other or non-DSP modules (such as the host interface) in an efficient, memory-mapped technique.

Each board contains four TMS32020 digital signal processor modules; a host computer interface, and an input/output bus for external data acquisition (see Figure 3-1). The DSPs and host interface are linked with a high-speed, parallel bus, called the Odyssey bus. This bus consists of a 24-bit address line and a 16-bit data line, with 2-Mbyte address space allocated to each board. The Odyssey bus can also be extended off-board to tie multiple Odyssey boards together in a large extended memory fashion. All memory, command, and interrupt functions are memory-mapped across boards. Hence, any DSP can access any other DSP’s memory as an extension of its own data memory. In effect, any processor's local memory becomes global memory to other digital signal processors. With the interrupts and commands also memory-mapped, a peer relationship exists among the DSP
devices such that any DSP can interrupt or pass commands to any other DSP.

![Diagram of a Parallel Signal Processor System (Odyssey)](image)

Figure 3-1 A Parallel Signal Processor System (Odyssey)

The data rate on the Odyssey bus is 650K words (16 bits) per second. With fair arbitration (no module can dominate) and only single-word transfers allowed for typical module/module or host/module transfers, an approximate linear relationship exists between individual module bus access rates and the number of modules continually contending for the bus. In other words, the worst case bus data rate available for module 4 would be approximately 163K words per second. Of course, bus activity varies greatly, depending on DSP algorithm requirements to exchange data between modules.
The Odyssey architecture overcomes several limitations of other multiple DSP designs. Since all of each DSP's memory is available to any other DSP, the access limitations of dual-port memory to additional processes or other specialized common memory are overcome. Also, since each processor can initiate a global memory access, the number of DSPs that can communicate with each other is not constrained. The number of processors can be increased by adding more boards as application requirements grow. The modular nature of the architecture greatly eases the task of upgrades to newer and faster DSPs.

The Odyssey architecture lends itself to a variety of signal processing problems and offers the following advantages and features:

- Sufficient processing speed for many real-time DSP problems. Specifically, the system allows 20 million multiply/accumulates (MAC) per second per board.
- Modularized algorithms readily map into the architecture for concurrent execution.
- Ease of programming allows rapid change of the modularized portions of the algorithms.
- The modularized algorithm sections may exchange data in a flexible manner.
- 512K bytes of data space are available per board.
- A high-speed data input/output channel is available on each board.
- An efficient link to the host is available to integrate symbolic processing in a problem solution.

\(^1\) We have recently upgraded the laboratory version Odyssey board with the CMOS, 100-ns cycle time, digital signal processor TMS320C25, and faster memory so that 2x speed can be achieved.
3.4 Grammar-Driven Connected-Word Recognizer (GDCWR)

Figure 3-2 Block Diagram showing the components of the Grammar-Driven Connected Word Recognizer

Figure 3-2 shows a block diagram of the GDCWR system. An isolated recognizer outputs all the words that are hypothesized along with their corresponding distance scores and estimated durations. The basic technology of the word recognizer is a modification of the original Texas Instruments LPC-based isolated word recognition system. The sentence hypothesizer constructs probable sentences from the word hypotheses and their time marks, and invokes grammatical constraints to consider only the admissible paths to output a recognized sentence with the lowest distance score. The list of possible sub-sentences is pruned to minimize both memory and processor requirements. The distance measure for the sentence has three parts: the first component is the sum of individual word distance scores multiplied by corresponding word durations; the second is a penalty for overlap or under-lap or adjacent words; and the third is a silence (null speech) distance measure. An important feature of the recognizer is that the sentence hypothesizer does not control the isolated word recognizer by any feedback. This ensures that all information is preserved for late binding and possible recovery from higher-level errors.
The word hypothesizer in CWR consists of two parts, a preprocessor, and a word hypothesis generator module. The preprocessor operates only on the speech input, characterizing it as a time-sequence of feature vectors. A linear prediction analysis is performed and transformed to an appropriate feature vector every 20 ms. The word hypothesis generator module compares the feature vector sequence of the input with the reference templates using a dynamic time warping algorithm, and outputs those words (along with their associated information) that are hypothesized to be close to the input speech. The number of word comparisons that can be performed is limited by the processing capability of the processing system.

The word hypothesis generator is computationally intensive and consumes more than 50 percent of the total processing time on a single processor. When these three modules, the Preprocessor, the Word Hypothesizer, and the Sentence Hypothesizer, are all implemented on a single processor, the vocabulary size is only about 30 words. Loading varies because the Sentence Hypothesizer is not fixed and is a function of the size and complexity of the grammar and the input utterance even though the loading of the word level similarity measurements is a fixed, predictable function of the vocabulary size. In a multiple processor environment, one or more processors can be completely allocated to word hypothesis generation, with another processor doing sentence hypothesis, and yet another one doing preprocessing. Thus, the vocabulary size can be increased by duplicating the Word Hypothesizer on more than one processor with each processor addressing a smaller subset of the vocabulary. The parallelism offered by the Odyssey architecture lends itself very well to this application. We have developed a single-board implementation of CWR with two processors devoted to word hypothesizing with a total vocabulary size of 100 words. Additional boards are

\[\text{The limitation is on the total number of reference frames of data that can be handled by each processor before it runs out of real-time operation. At 8 kHz, this limit is 540 reference frames per processor with a frame period of 40 ms.}\]
capable of handling 200 words each, where all four processors can be devoted to word hypothesizing function.

3.4.1 Task Partitioning and Allocation

![Diagram of Task Partitioning](image)

Figure 3-3 Task Partitioning

In an ideal multiple processor environment one would expect the throughput of the system to increase linearly as the number of processors increases. However, this is not always true. In practice, the throughput in a multiple processor system increases significantly only for the first few additional processors and in fact begins to decrease after a certain number of processors. This is due to increased inter-processor communication (IPC). This occurs when software modules, resident on different processors, need to communicate with each other. Communication protocols, management of storage, waiting time in queues etc. all

which for 25 reference frames per second of speech translate to 50 words per processor
contributes to the overhead. This overhead grows rapidly with large numbers of highly interacting processors and the system throughput actually begins to decrease. This is referred to as the saturation effect as shown in Figure 3-4.

![Figure 3-4 System Performances on Multiprocessor System](image)

The designer is now faced with a dilemma. In order to exploit the computing resources, offered by the multiple processor system he needs to balance the load. However, balancing the load creates inter-processor overhead, which needs to be kept as low as possible. One way to compromise these two conflicting factors is to allocate closely related software modules to the same processor and keep the communication between processors to a bare minimum. This demands a thorough understanding of the algorithm and the flow of data involved.

The first step is to partition the algorithm into several individual sub-tasks or modules. For example, GDCWR has been split up into several subroutines, which have been arbitrarily named A, B, C, etc. Subroutine A calculates the 11-autocorrelation values from a frame of digitized speech samples; B is a routine that computes the reflection coefficients and so on. These subroutines, represented by circles, are shown in Figure 3-3 and are connected in accordance with the flow of data. The number of words...
being passed from one sub-routine to another on a per frame basis represents the inter-module communication and has been placed on the connecting arcs. This process is known as task partitioning.

Once the task partitioning is completed, the next step is to allocate these modules to different processors so that the system throughput is maximized. This is task allocation. It is during this phase of the design that one has to balance the two conflicting factors of load distribution and minimum inter-processor communication. To maximize throughput, the individual processors should be able to run autonomously to the extent possible.

The first step in task allocation is to identify those routines that are closely related and/or communicate with one another extensively. In Figure 3-3 routines A, B, C, E, F and I are closely related and therefore fused together to form a bigger module called the Preprocessor which is allocated to one processor. It was found that H contributes to more than 50% of the loading and limits the vocabulary size. An entire processor must therefore be devoted to doing H. However, there is considerable traffic between H and P and inter-processor communication would be increased if these routines were resident on different processors. H and P are therefore fused together to form a bigger module called the Word Hypothesizer and allocated to another processor. U is a routine that could be allocated to the Word Hypothesizer or the Preprocessor, but since we wish to allocate as much CPU time to the Word hypothesizer as possible to do the similarity measurements, U is allocated to the Preprocessor. The remaining routines G and S comprise the Sentence Recognizer and are allocated to another processor. This completes the task allocation of the CWR software. The basic recognition system therefore requires three processors viz., the Preprocessor, the Word Hypothesizer and the Sentence Hypothesizer. The parallelism offered by a multiprocessor architecture can now be utilized to increase the active vocabulary size by the concurrent execution of the Word Hypothesizer on two or more processors with each processor addressing a smaller subset of the vocabulary.
Figure 3-5 shows the allocation of tasks to different processors on one Odyssey board. Processor 0 is the Preprocessor, Processors 1 and 2 are the Word Hypothesizers, and Processor 3 is the Sentence Hypothesizer. Note that all word hypothesizers operate on the same data from the preprocessor, and communicate with a single sentence hypothesizer. A single Odyssey is capable of recognizing about 100 words. Each additional board is capable of addressing 200 words each.

In designing real-time systems, one tends to optimize the entire software. Optimization of real-time software, though desirable, may not necessarily be practical. The resulting increase in processing efficiency does not justify the effort required to optimize all the code. It is often found that there are only a few sections of code where a large percentage of the total processing time is spent. Hence, efforts should be directed towards optimizing only these small sections of the code. For example, in the Word Hypothesizer module, it was found that 90 percent of the time was spent in the distance measuring routine that compared the input speech with the
stored references. Consequently, only this module was optimized. Multi-processor software should be designed so that it can be easily debugged. As with most computer systems the design and specification of a multiprocessor system is done top-down and debugged bottom-up. Thus it is important that one be able to debug the module associated with each processor individually. In the GDCWR implementation, each processor module is designed to communicate via I/O buffers. During the debug process, the input buffer is filled with canned data and the processor is made to execute its function. The output buffer can then be examined for correctness. Using this technique each processor module can be tested prior to integration of the entire application.

3.4.2 Performance Testing

The performance of a connected word recognizer is extremely difficult to quantify because of the lack of accepted database and measurement standards. However, Texas Instruments has done a limited amount of testing on this algorithm using an internally developed connected digit database. The data used to test the algorithm consisted of 20 speakers reading 5-digit strings. A total of 2000 strings was tested. Two application scenarios are of interest - those applications where the length of the digit sequence is unknown and those (like telephone numbers for example) where the length of the sequence is known. The results of the test are summarized below:

<table>
<thead>
<tr>
<th>Length</th>
<th>Unknown Length</th>
<th>Known Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.2% sentence error rate</td>
<td>3.4% sentence error rate</td>
</tr>
<tr>
<td></td>
<td>1.1% word error rate</td>
<td>0.7% word error rate</td>
</tr>
</tbody>
</table>

Note that the word error rate for digit strings of known length approaches that achieved for the best-isolated word systems.
3.4.3 Conclusions

This chapter presented the issues involved in partitioning and allocating tasks in a multiprocessor environment and discussed in detail the implementation of a connected word recognizer on a novel coarse-grain computer architecture involving multiple digital signal processors with a symbolic computer host providing an intelligent signal processing environment called the Odyssey/Explorer system. Each word hypothesizer is capable of addressing about 50 words providing a 100-word capability for the first Odyssey board and 200 words for each additional board.