Implementation of an HMM-Based, Speaker-Independent Speech Recognition System on the TMS320C2x and TMS320C5x

B. I. (Raj) Pawate
Peter D. Robinson
Speech and Image Understanding Laboratory
Computer Sciences Center
Texas Instruments Incorporated

Abstract

In the years to come, speaker-independent speech recognition (SISR) systems based on digital signal processors (DSPs) will find their way into a wide variety of military, industrial, and consumer applications. This paper presents an implementation of a hidden Markov model (HMM) speech recognition system based on the 16-bit fixed-point TMS320C2x or TMS320C5x DSP from Texas Instruments. It includes a description of a minimal TMS320C5x-based system and shows how the HMM algorithm and the system algorithm interact. The report also presents system loading, along with a current list of the speech templates. In addition, it presents data showing the relative performance of the algorithm in a controlled environment. A discussion is included on future algorithm enhancements and reduction of the physical hardware system. The paper concludes with a description of a very large vocabulary SISR system based on the TMS320C3x and TMS320C4x floating-point DSPs.

Background

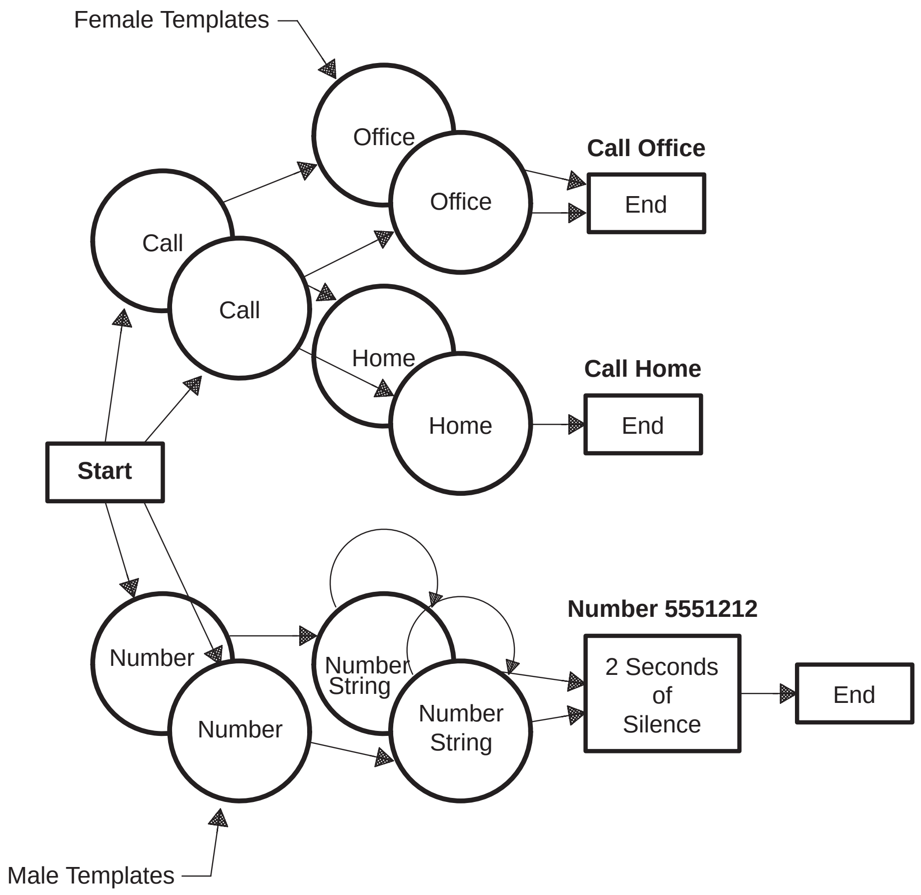
Prior to the introduction of the HMM word recognizer, speech recognition was based almost entirely on dynamic time warping (DTW) techniques. These DTW recognizers are limited in that they are speaker dependent and can operate only on discrete words or phrases (pseudoconnected word recognition). They employ a traditional bottoms-up approach to recognition in which isolated words or phrases are recognized by an autonomous or unguided word hypothesizer. The recognition technique employed in DTW is straightforward. A set of speech templates are maintained in memory for each word/phrase in the vocabulary. These templates are based on linear predictive coding (LPC) models. As a new word or phoneme is acquired and placed in the speech queue, its features or characteristics are compared to each memory-resident template one word/frame at a time. As the acquired speech frame is compared, it is stretched or compressed in time to optimize the correlation between the memory-resident templates and the queued speech frame (hence, the term dynamic time warping). As this warping process progresses, the result of the optimized correlation is logged, and the score is updated. This is repeated for each template in the current vocabulary list. A process is then run on all the collected scores, yielding a best-guess estimate or hypothesis of the recognized word. These DTW systems have been implemented on DSP platforms with a throughput as small as five million instructions per second (MIPS).

The TMS320-Based HMM Recognizer

The Texas Instruments speaker-independent continuous word recognizer provides a top-down approach to speech recognition using the continuous-density hidden Markov model. The Markov model (or the Markovnikov rule) was introduced by a Russian organic chemist, Vladimir Vasilyevich Markovnikov in 1870. HMMs are statistical or stochastic processes, which, when applied to speech recognition, bring machine-based voice recognizers to new levels of performance. However, this increase in performance has its price. HMM-based speech recognizers require a digital signal processor, such as the TMS320C25, that can execute a minimum of 10 MIPS. As this report shows, the improved accuracy and system flexibility provided by the HMM-based system outweighs the added cost of the ’C25 or ’C5x over the ’C1x (5 MIPS).

The Texas Instruments Speech Research Group in Dallas implemented the HMM-based speech recognizer described in this paper on a ’C25 in June 1988. This original application, which contained a vocabulary of 15 words (15 male and 15 female templates), implemented a voice dialer to show proof of concept. Figure 1 shows the grammar rules or vocabulary flowchart for this application.

Figure 1. Voice Dialer Sentence Hypothesizer Flowchart



The HMM voice dialer can currently run on three platforms:

- A stand-alone TMS320C25-based voice dialer demonstration box
- A custom dual TMS320C25-based development platform named *Calypso*
- The TMS320C5x Evaluation Module (EVM) with analog front-end board

In addition to the 15 words used in the voice dialer application, a total of 49 voice templates (male and female) are available for a user’s unique end application. Table 1 lists the 49-word HMM vocabulary. Example sentences follow the table.

Table 1. Current HMM Vocabulary (49 Words)

ADD	AREA_CODE	BACK	BLOCK
CALL	CANCEL	CONFERENCE	CREATE
DELETE	DISABLE	DISTURB	DO
EMERGENCY	ENABLE	ENTER	EXTENSION
FORWARD	FROM	HOLD	HOME
LAST	MAIL_LIST	MESSAGE	NO
NOT	NUMBER	OFFICE	PLAY
PROGRAM	RECORD	REDIAL	REVIEW
SEND	STOP	TO	TRANSFER
WAITING	YES	ZERO	OH
ONE	TWO	THREE	FOUR
FIVE	SIX	SEVEN	EIGHT
NINE			

Example sentences from the vocabulary include:

- Call home
- Call office
- Call number five five five one two one two send
- Call extension two three enter
- Delete extension three five enter
- Create extension seven seven enter
- Disable do not disturb
- Disable call waiting
- Enable call back
- Block last call

Voice-Dialer Performance Testing

In 1990, a test was conducted on the HMM recognizer using the standalone voice-dialer demonstration box. A total of 2,272 sentences were tested in a closed-set experiment utilizing word templates from the vocabulary database noted above. Test sentences included the words *call*, *office*, *home*, *area_code*, *number*, *extension*, *enter*, and *cancel*, in addition to the normal digits. Speed-dialed sentences included the words *home*, *office*, and *emergency*.

Sentence Recognition Performance

Total number of sentences	2,272	
Total sentence errors	133	(5.9%)
With substitutions	73	(3.2%)
With deletions	41	(1.8%)
With insertions	19	(0.8%)

Word Recognition Performance

Total number of words	12,785	
Total word errors	148	(1.2%)
With substitutions	84	(0.7%)
With deletions	45	(0.4%)
With insertions	19	(0.1%)

System Considerations

The system considerations or desirable objectives for a recognizer can be broken into two categories — functional (or ergonomic) and technical:

Functional Requirements

- Speaker-independent recognition (no training required)
- Recognition of connected or continuous words (natural speech)
- High level of accuracy
- Ability to work on a wide cross section of dialects
- Reasonable size of vocabulary
- Affordability

The functional criteria are straightforward, and the system should perform well enough to make it usable in a quiet environment by a majority of the population. Current low-cost, machine-based recognizers are not sufficiently robust to recognize all people at all times. Therefore, it is important to set limits on the level of performance. These limits or restrictions can be determined only through experimentation and test marketing.

Technical Requirements

The technical objectives are much easier to define because they are price- and performance-driven.

- Utilize as little memory as possible
- Work on a 16-bit fixed-point microcontroller/DSP
- Incorporate minimal chip count for a small system form factor
- Use single voltage and low power for battery operation

Things to Come

As the technology progresses, speech recognition will find its way into a wider base of applications. These developments are currently under way at Texas Instruments:

- Adaptation of a microphone array for acoustic beam forming
- Active creation or modeling of background noise for noise templates
- Speaker-adaptive speech recognition
- A mix of speaker-dependent and speaker-independent recognition

Each of the listed techniques may or may not increase the perceived performance. However, they all show promise. The hardest problem to overcome is background noise management, or the art of listening in the presence of noise. Noise management algorithms require an extensive amount of processing power to implement. As an example, adaptive noise cancellation deals with the problem of removing correlated noise (that is, noise that has some redundancy associated with it). This process requires large amounts of data memory, and, as noted, it is computationally intensive. Another technique that shows promise is the use of a microphone array. The array can focus or listen in a specific direction while subtracting the noise in all other directions. Another noise-related enhancement is the real-time creation of a template that matches the current background noise. This technique tries to cancel noise by ignoring it; hence, if the noise is known, a null set is returned when the noise is detected.

In addition to enhancing noise performance, it is also desirable to increase the flexibility of the machine-based recognizer. One technique currently under development at Texas Instruments is the inclusion of a speaker-adaptive algorithm. In this algorithm, the SISR routine comes with a set of general-purpose RAM-based templates that are initialized during runtime from some nonvolatile storage media. As a user interfaces with the machine, the machine modifies or optimizes the templates for that user. This technique is useful when there is only one user per session, such as with a PC-based SISR system.

In the area of speaker-dependent and speaker-independent recognition, a provision will be made so that a user *can* supplement the existing speech library with user-recorded templates, such as a trade or personal name: for example, CALL JIM.

Example Platform

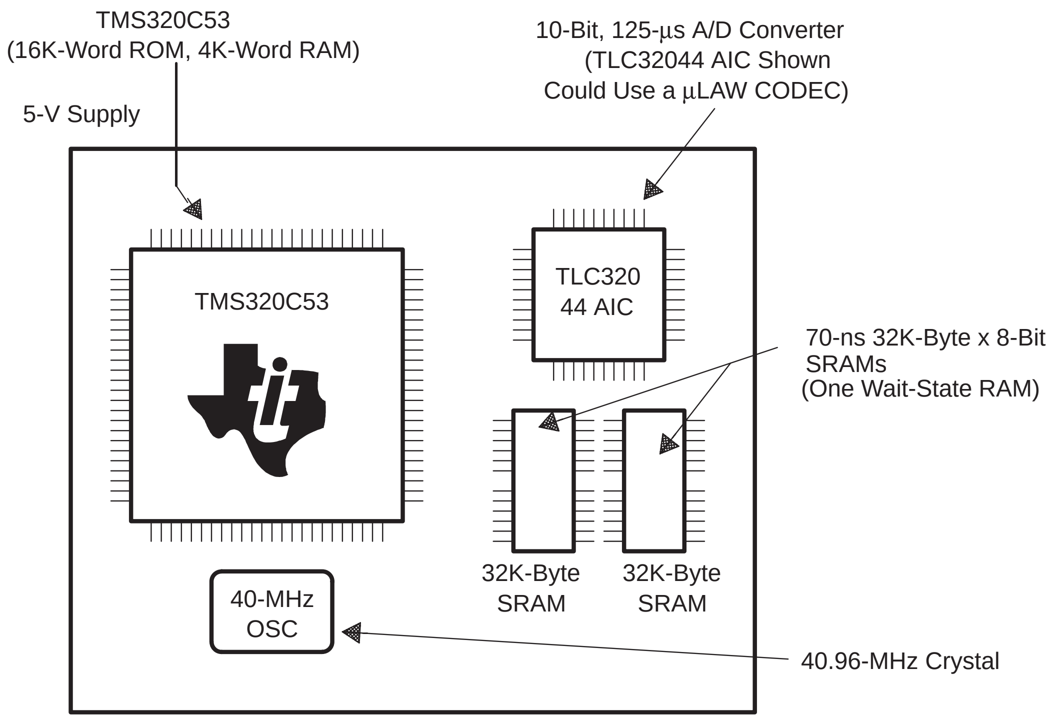
The current fixed-point HMM recognizer, running the voice dialer vocabulary shown in Figure l, requires a little over 6K words of program and around 40K words of data memory. Table 2 breaks out memory loading on a module-by-module basis and reflects performance on a ’C5x platform running at 20 MIPS.

Table 2. HMM Processor Loading on a TMS320C5x

Module	Data Memory (Words)	Program Memory (Words)	CPU Loading at 20 MIPS
Feature Extractor	5K	1.8K	7%
Compute Word	16K + 0.75K/word (est.)	0.6K	21%
Compute Sentence	5K	1.2K	12%
HMM Executive	0.1K	1.8K	7%
Initialization and I/O	0.1K	0.5K	2%
Totals	**26.2K + Compute Word Templates**	**5.9K**	**49%**

Given the total system memory requirements, this algorithm could be packaged in a single ’C53 with one external A/D converter and two external 70-ns 32K-byte $\times$ 8-bit SRAMs. Note that the entire program memory (5.9K words) can reside in ROM. However, all data memory except the compute word templates (0.75K bytes $\times$ 16 bits per word) must be of the read-write type.

Figure 2. A Minimal TMS320C53 HMM System



The system shown in Figure 2 provides l6K words of program ROM and up to 36K words of data RAM (it is assumed that there is a host interface for template upload; if not, an additional 1M words of ROM is needed). Further integration is possible with Texas Instruments customized DSP (cDSP) devices. A cDSP implementation will reduce this design to two chips: a monolithic DSP, including an A/D converter with system interface logic, and an external 70-ns 32K-byte $\times$ 16-bit SRAM (1/2M word SRAM).

How the Texas Instruments HMM Implementation Works

The Texas Instruments HMM speech recognizer consists of three computational processes running together: a feature extractor, a word hypothesizer, and a sentence hypothesizer. The feature extractor, as its name implies, reduces the continuous speech to a series of 20-ms frames or states whose features are reduced to a finite feature set called a generalized set feature, or GSF. The HMM processes *compute word* and *compute sentence* guide the recognition — first at the sentence level, then at the word level. These three processes interact so that the feature extractor feeds the word hypothesizer, which is no longer autonomous, but guided by a sentence hypothesizer. Hence, recognition is now accomplished on a state-by-state basis.

The HMM processes, at any level, *can be* expressed in terms of mathematical probabilities as the likelihood that one state follows another. If the vocabulary is known and the sentence structure is known and finite, then it is a simple process to predict the next state, given the present and past states. This is done by scoring frames of extracted features along paths that terminate at unique solution end points. Hence, paths scored at the state level point to word level, which points to sentence-level solution sets. All along the way, probabilities are calculated and assigned in guiding the process.

Figure 3 shows graphically how the HMM word hypothesizer works. Within the voice dialer system, after the word CALL is recognized, the sentence hypothesizer has only two paths from which to select: HOME or OFFICE. The lower portion of Figure 3 shows the path selection resulting in OFFICE.

Figure 3. Example of an HMM Flow

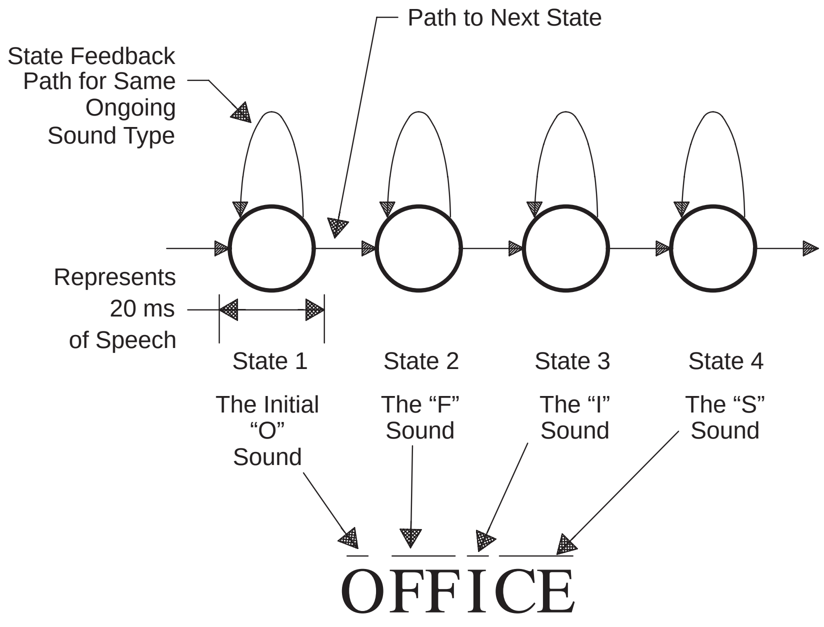


Figure 4 on page 9 shows how this application of the hidden Markov model continuous word recognizer is implemented on the ’C2x or ’C5x. The speech data flows through the model from left to right, while the recognition is driven from right to left.

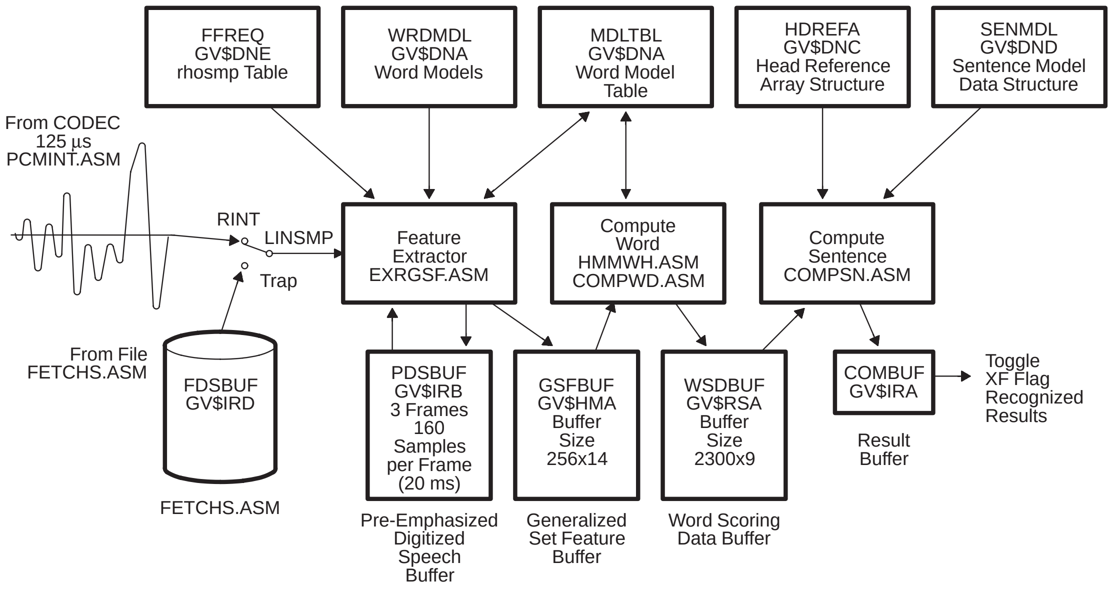
The process is started and sustained as follows. Time samples, which are taken every 125 µs, are queued in the pre-emphasis digitized speech sample buffer (PDSBUF). These samples are then operated upon by the feature extractor on a frame-by-frame basis (a frame is equal to 160 samples). The feature extractor interfaces to five data structures:

- The LPC filter coefficient table, or rhosmp table, as noted in Figure 4
- The pre-emphasis data structure buffer
- The word models
- The word model table
- The generalized set features buffer (GSFBUF)

These data structures and their contents are discussed on the following pages. In general, the feature extractor performs two functions:

1. It reduces a frame of speech data to a finite data set that describes the speech type. This reduced data is called a state, which is the smallest unit of time in the algorithm (20 ms).
2. Next, it expresses the state so it can be approximated by a Gaussian distribution.

Figure 4. Block Diagram of the HHM Recognizer

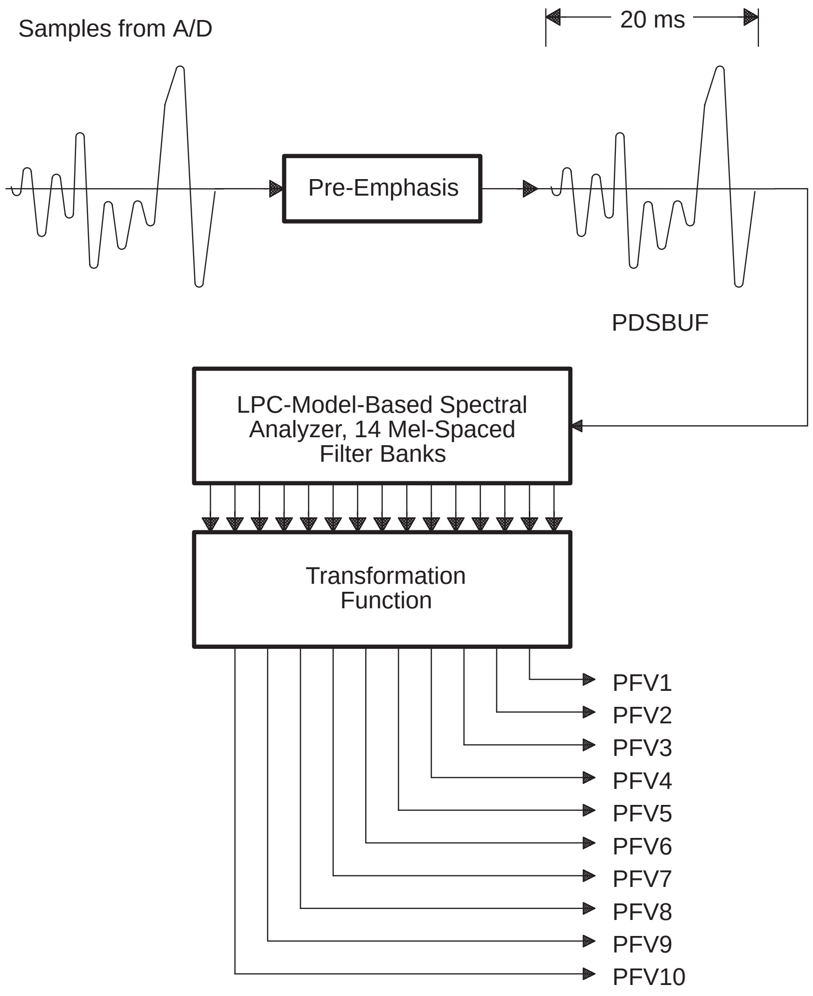


Once a frame of speech is processed by the feature extractor, the results are queued in the GSFBUF. At this point, the 160-element frame has been reduced to a 14-element state vector in the GSFBUF. These 14 memory elements contain the following information:

- Frame energy
- Inner product
- 10 principal feature vectors (PFVs), PFV1 through PFV10
- Utterance flag (voiced or unvoiced speech)
- Ongoing-word flag (still in same utterance)

Once a new frame is added to the GSFBUF, the HMM process takes over (compute word and compute sentence). The function of the HMM is to present a hypothesis on an optimal path for the frame. Hence, the contents of the GSFBUF are continuously being interrogated by the word hypothesizer to determine the best path score to a unique end point (word), given the current state and previous states observed. In addition to the RAM-based buffers, there are three ROM data structures that the feature extractor accesses. The rhosmp table contains all the coefficients used in the various data reduction routines within the feature extractor, the 14 5-tap filters, and the LPC-10 coefficients.

Figure 5. The Feature Extractor



Next, the word model (WRDMDL) data structure contains all the word model templates in the vocabulary. This buffer is typically the largest memory array within the recognizer. The word hypothesizer indexes into this data structure via the word model table (WDLTBL). This table contains the starting address, length, and word ID for each word model. As noted above, extracted features or states are queued in the GSFBUF. They are correlated against the valid word models, as determined by the word and sentence hypothesizer for that state.

Once a word is processed, all associated state vectors are removed from the GSFBUF and transferred to a slot of a buffer in memory called the word scoring data buffer (WSDBUF). Each slot in the WSDBUF stores the following:

- Level index — sentence-level, word-level, or states
- Model index — current index pointer into the word model data structure
- State index — what index within the model
- Path_scr — best-path score for the current frame
- Path_ptr — scoring data structure (SDS) index of the previous frame in best path
- Time — input frame time index
- Last_time — time index of last path through this point
- Next_state — SDS index of next state for this model
- Next_word — SDS index of next word

The data is now reduced from 160 words (PDSBUF) to 14 words (GSFBUF) and to a 9-word WDS buffer. However, as multiple state vectors were required for each word in the GSFBUF, multiple WDSBUF frames are needed for best path determination, resulting sometimes in an increase in total memory requirement to sustain the HMM process. The final phase in the HMM process is the reduction and subsequent linking of the WSDBUF vectors (based on their path score) so that an optimal path vector set remains. This vector set points to a unique word ID determined by the read-only sentence model (SENMDL) data structure. This array maps vocabulary words to model IDs within the sentence IDs or compute sentence, in conjunction with the head reference array structure (HDREFA). This read-only array maps a vocabulary word ID to its first model ID. The subsequent fields in the HDREFA model table are used to link multiple models for a given word ID when that word ID is used more then once in a sentence model. With the IDs known, the word IDs are passed to the COMBUF, where the host system reads the results. Hence, the COMBUF contains the recognized words that are to be returned to the host. The COMBUF is organized as follows:

- ID of the recognized word model
- The error (32 bits) in the word model
- Frame index of the word model’s beginning
- Frame index of the word model’s ending
- Frame index at which the word model was created

Fixed Point Versus FloatIng PoInt

Thus far, the discussion has focused on implementing the HMM algorithm on a fixed-point DSP. A floating-point processor such as the TMS320C3x, with its vast 16M-word address range, DMA controller, and inherent floating-point attributes, makes coefficient representation a nonissue. The elimination of numerical concerns *can* significantly reduce development time, but this is not necessary for implementing the HMM algorithm. As shown, a fixed-point processor performs the algorithm equally well and can significantly reduce system cost. However, executing a fixed-point system requires a thorough understanding of the complex numerical issues. Typical fractional variables, such as the features used to represent the acoustic data (GSFBUF), are represented on a fixed-point DSP by using a $Q_{n/m}$ format. With this format, the 16-bit 2s-complement field is evaluated with a sign bit, n integer bits, and 15-n fractional bits.

Figure 6. Example of Q $_{4/16}$ Notation

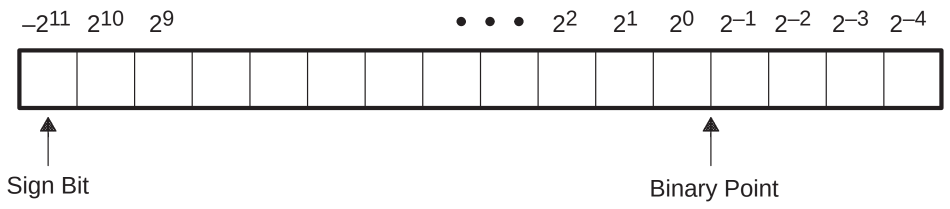


Figure 6 shows the dynamic range of a $Q_{4/16}$ number is: $2^{11} - 2^{-4} = 2047.9375$ to $-2^{11} = -2048$, where a $Q_{15/16}$ number would range from $2^{0} - 2^{-15} = 0.99996948$ to $-2^{0} = -1$.

Table 3 shows several examples of $Q_{n/m}$ notation, as used in the implementation of the fixed-point Texas Instruments HMM recognizer.

Table 3. Examples of $Q_{n/m}$ Notations (Fixed-Point Representation)

Variable	$Q_{n/m}$ Notation
Feature vector	$Q_{4/16}$
Cumulative pathscores	$Q_{15/16}$
Log of transition probabilities	$Q_{15/16}$
Log of observation probability	$Q_{15/16}$

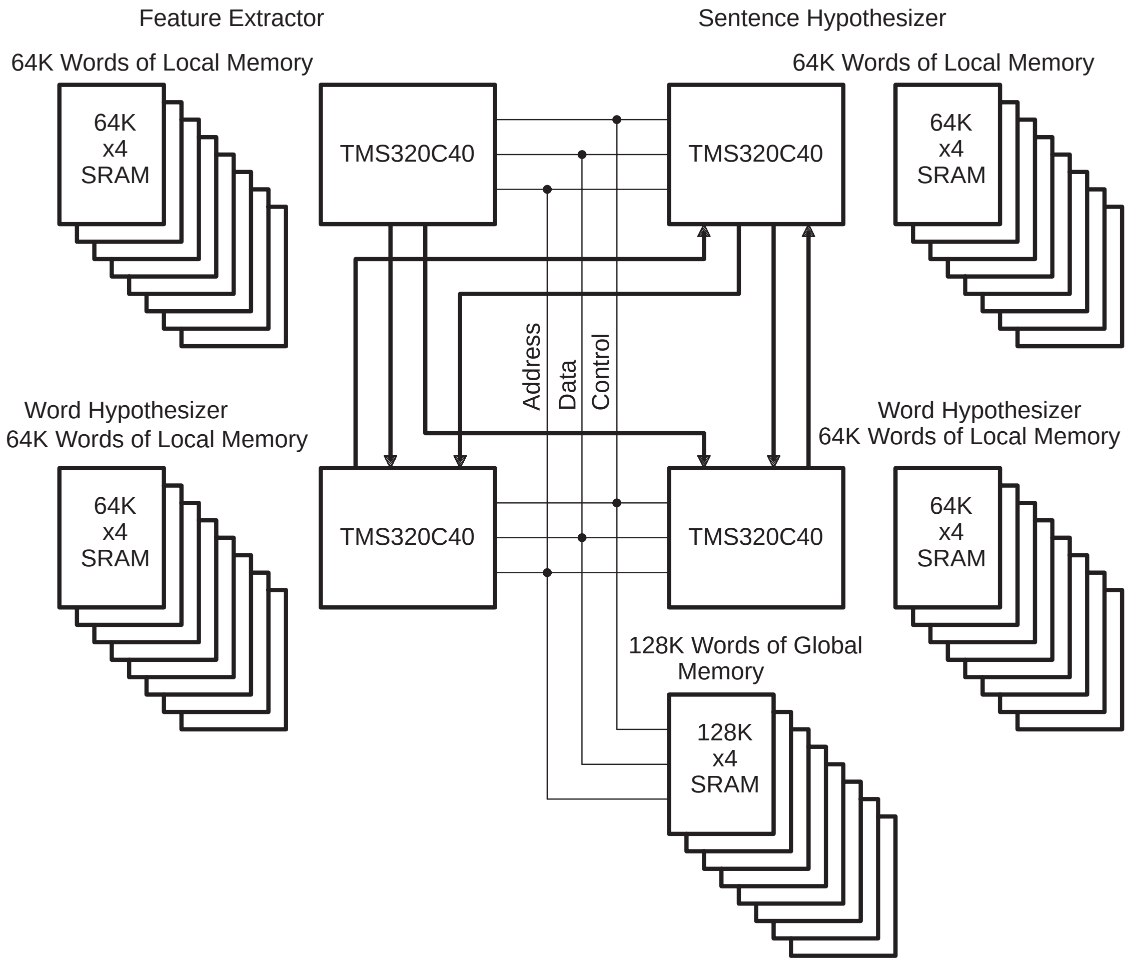
As noted, a fixed-point DSP can reduce system cost. However, in SISR systems, size of vocabulary is all-important. SISRs need a floating-point system, not only for its ability to represent values in floating-point format, but also for its memory reach. The ’C2x and ’C5x can access only 64K words of linear data memory, while the ’C3x can access 16M words, and the ’C4x 4G words. The size of the vocabulary is limited only by processing power, as opposed to accessible system memory. To implement a very large vocabulary recognition system via the HMM technique presented here, the following must be accomplished.

- The feature extractor must be improved to increase its granularity and to make it more robust. As more and more words are added to the vocabulary data base, it becomes increasingly more difficult to distinguish similar sounding words.
- A sentence hypothesizer must be developed that can track and predict words according to grammar rules for the English language. In addition, the sentence hypothesizer must be adaptive in that it must be able to learn user-specific grammar rules (slang).
- A word hypothesizer must be developed that is speaker adaptive (work ongoing) and allows the addition of user-defined vocabulary (again, work ongoing).
- A technique must be developed for creating templates from text-based descriptions. Optimally, these descripters should be based on a published standard, such as the symbols used in the respelling for pronunciation, as found in dictionary pronounciation guides.

Example: **elephant** _(ĕl′ə -fə nt)

Figure 7 shows a very large vocabulary SISR system based on the 'C4x parallel processor development system (PPDS).

Figure 7. SISR System for Very Large Vocabulary


Feature Extractor
Sentence Hypothesizer
64K Words of Local Memory
64K Words of Local Memory
64K x4 SRAM
TMS320C40
TMS320C40
64K x4 SRAM
Address
Data
Control
Word Hypothesizer 64K Words of Local Memory
Word Hypothesizer 64K Words of Local Memory
64K x4 SRAM
TMS320C40
TMS320C40
64K x4 SRAM
128K Words of Global Memory
128K x4 SRAM

The feature extractor, compute sentence, and word hypothesizer are distributed over the four 'C40s. The word hypothesizer uses two 'C40s because it is the most computationally intensive task. The feature extractor feeds output (frame or state data) to the two word hypothesizers via two 8-bit parallel ports. In addition, the sentence hypothesizer feeds both word hypothesizers, which, in turn, feed their results back to compute sentence. Although the above system has not been implemented, it demonstrates a logical progression of the technology.

Conclusion

In summary, one TMS320C53 DSP can implement a robust HMM speaker-independent speech-recognition system with just under 50% processor loading. This, with future enhancements to the existing HMM SISR algorithm and hardware systems, makes a single-chip DSP-based recognizer in a noisy environment a reality. This paper discusses the system resource requirements, vocabulary flexibility, and possible future enhancements. The data presented shows how a fixed-point processor is ideal for small vocabulary systems in which expense and power are a concern.The paper also shows how this HMM-based algorithm can be adapted to a floating-point processor, allowing for a very large vocabulary system.