SPEECHCORDER, THE PORTABLE MEETING RECORDER

Adam Janin and Nelson Morgan

International Computer Science Institute
1947 Center Street
Berkeley, CA 94708
janin@icsi.berkeley.edu morgan@icsi.berkeley.edu

ABSTRACT

SpeechCorder is a project to design, implement, and test a portable speech recognizer. It will be used to retrieve information from roughly transcribed speech recorded during natural meetings. This is an important application domain, but has inherent difficulties far beyond the command-and-control functions that are beginning to be implemented with speech recognition on portable computers. In addition to the difficult acoustic environment presented by natural meetings, the limitations of a portable platform must also be considered.

The proposed system uses IRAM, a new chip being developed at the University of California, Berkeley. It is a low-power, vector processor with embedded DRAM. By coding the speech recognition algorithms to take advantage of IRAM, high performance with low power consumption can be achieved.

In this paper, we will describe the proposed SpeechCorder system, some of the research issues and design tradeoffs, and the status of work in progress.

1. SPEECHCORDER APPLICATION

The goal of the SpeechCorder project is to produce a portable device that records meetings in real-time and generates a searchable transcript. Users would annotate and correct the transcripts using voice and pen input. They would also retrieve records based on spoken and textual queries.

1.1. Why Not Written Notes?

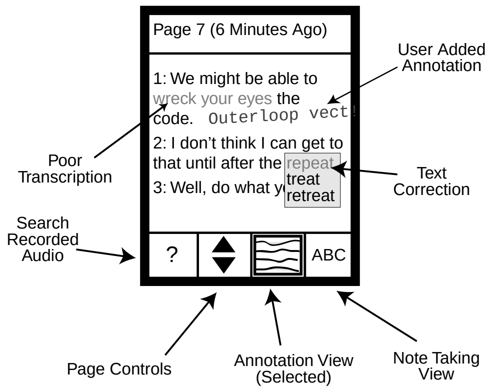
- Unreadable Handwriting: Often, written notes are illegible. This becomes even more of a problem as time passes — notes that were comprehensible when written become indecipherable after a few weeks.
- Detracts From Listening: It can be very difficult to take notes and pay close attention to the meeting at the same time. Many people find that they can take notes or they can listen closely, but not both.
- Incomplete Record: It is almost impossible to record a meeting in its entirety. Frequently, important points and data are missing from the written record.
- Hard to Search: Written notes provide no support for searching and indexing. Finding particular information is very difficult.

1.2. Why Not A Tape Recorder?

- Hard to Annotate: With written notes, adding arbitrary annotations (underlining, circling, diagrams, doodles) is trivial. It is much more difficult to provide synchronized annotations with a tape recorder.
- Playback Can Be Disruptive: One cannot play back a tape recording during a meeting without disturbing the other participants. Also, one cannot play back and record simultaneously.
- Too Much Data: Recording an entire meeting via tape produces a very large amount of extraneous data.
- Hard to Search: As with written notes, a taped meeting cannot easily be indexed and searched.

1.3. Hypothetical SpeechCorder Example

Figure 1 shows a hypothetical screen of the SpeechCorder application. The image demonstrates some of the potential user-interface ideas for correction and annotation of the transcript.


Page 7 (6 Minutes Ago)

1: We might be able to wreck your eyes the code. Outerloop vect

2: I don't think I can get to that until after the repeat treat retreat

3: Well, do what y

Figure 1: Hypothetical SpeechCorder Screen Shot.

As participants of the meeting speak, a possibly inaccurate transcript would appear in real-time. Since the speech recognizer often can produce a measure of how confident it is with its guess, the user-interface could highlight poor transcriptions. When the recognizer makes an error, the user could quickly correct it by clicking on the incorrect word. The recognizer would then generate a popup box of alternatives from which the user could select the correct word.

Also shown is a text annotation entered by the user.

1.4. SpeechCorder Requirements

For SpeechCorder to be useful, it must meet a number of challenging requirements. It must record meetings in natural settings so that impromptu meetings can be processed. The speech recognition must therefore work in uncontrolled acoustic environments including background noise (fans, music, etc.), and reverberation using far-field microphones (e.g. a PDA microphone). The vocabulary must be large enough to cover the domain of the meeting. It must work with spontaneous speech.

If we want to support impromptu meetings in uninstrumented environments, it is necessary for SpeechCorder to be portable. Although it is certainly possible to use a hand-held computer as a terminal using a wireless network, we feel that a self-contained solution is better in the long run. The terminal/main-frame model has more components to fail — the terminal, the network, the wireless link, the main-frame, the infrastructure. There is also the question privacy, which is much easier to address with a self-contained unit. The PC revolution has also shown us the utility of *personal* compute power.

Perhaps the most useful aspect of having a transcript of a meeting is the ability to search the record. We plan to provide for searching both by speaker and text content. Since the actual audio will be stored, the users can play back the portion of the meeting that matches their criteria. We will support both textual and spoken queries.

Note that, since the audio record is stored, the transcript need not be perfect. It need only be good enough so that queries match where users expect them to match. For referring to the actual content, users can play back the audio. In addition, the speech recognizer can output not just the most likely transcript, but also a list of the top few most likely hypotheses (N-best lists). Queries can be made against these N-best list, rather than just the best hypothesis. Finally, the recognizer does much worse with so-called function words (such as "the", "a", "of", "an") as opposed to content words. However, for text retrieval, systems usually ignore function words. For all of the above reasons, it is perfectly satisfactory for the recognizer to be imperfect. In fact, we expect word-error rates of up to 40% to be acceptable for information retrieval [3].

2. IRAM

The SpeechCorder project is closely affiliated with the Intelligent RAM (IRAM) group at the University of California, Berkeley [6]. The IRAM group is building a powerful processor well suited to our task. In this section, we will briefly outline the architecture of the IRAM chip, and discuss its advantages and disadvantages. The following section will detail how ASR algorithms map to the IRAM architecture.

The IRAM architecture consists of a scalable design combining a vector processing unit with a DRAM array and fast I/O on a single chip. This allows efficient processing of multimedia data with fairly low power consumption. The architecture allows for considerable scaling in processing power and memory while maintaining instruction set architecture (ISA) compatibility. In the section below, we will discuss some details of the first implementation of IRAM, VIRAM-1.

2.1. Embedded Memory

The first distinguishing characteristic of IRAM is that the memory is incorporated directly on the same chip as the processor. Not only does this allow a high peak memory bandwidth of 25.6 GBytes/s, but it also reduces power consumption by removing the need to drive external busses and cache local data. A deep vector pipeline helps hide the latency of the DRAM.

Memory can scale up both in total amount, and also in the number of banks. More banks provide higher efficiency by allowing multiple memory accesses to overlap. Because of limitations inherent in an academic research project, we are constrained to use available memory "macros", which limit the number of banks in VIRAM-1. Also, because of packaging considerations (the chip has a high aspect ratio), total memory in VIRAM-1 is only 16 MBytes.

2.2. Vector Processor

In addition to embedded memory, IRAM also incorporates a vector coprocessor. The coprocessor executes a set of instructions based on an extension of the MIPS ISA to a vector register architecture. Vector instructions perform a set of identical operations on the elements of vector operands located in a vector register file. For example, an instruction such as **`vadd vr0, vr1, vr2`** would add all the elements in vector register `vr1` to the corresponding elements in vector register `vr2`, and store them in vector register `vr0`.

On VIRAM-1, each vector register consists of 2048 bits, and can be divided into 32 64-bit elements, 64 32-bit elements, or 128 16-bit elements. The ISA allows 258 8-bit elements, but 8-bit data types are not supported in VIRAM-1 (they will generate an unimplemented instruction exception). For vectors longer than the vector length, strip mining is required. The ISA defines instructions and registers that isolate the application from the details of the chip implementation. Therefore, the application need not be aware of the size of the vector register, only of the width of the data type. IRAM also operates very efficiently on short vectors.

VIRAM-1 contains 4 vector lanes, each containing 2 arithmetic units capable of operating on 64-bits at a time. Peak performance at 200 MHz is 6.4 Gops for 16 bit integer data, 3.2 Gops for 32 bit data, and 1.6 Gops for 64 bit data. Higher performance could be achieved by increasing the number of vector lanes. Since the ISA does not expose the number of lanes, programs need not be rewritten for higher performance IRAM chip implementations.

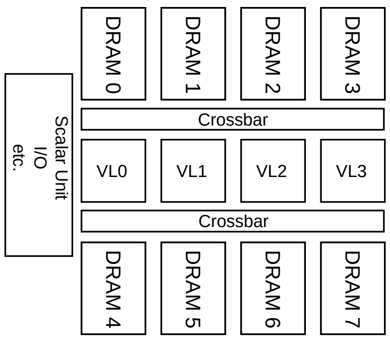


Figure 2: VIRAM-1 Floorplan

2.3. Advantages and Disadvantages

Instruction fetch, decode, and dispatch consume significant power in conventional architectures. Since a single vector instruction initiates many operations, far less power is used with a vector architecture. This, combined with the power savings from using embedded DRAM, leads to relatively low power consumption on IRAM. VIRAM-1 will consume approximately 2 watts of power, or more than 3 Gops / watt at peak performance.

Another key advantage of IRAM is that it is a general purpose processor. Although efficiency is certainly reduced if the code cannot be vectorized, it can still run. This allows us to concentrate effort only on the computational kernels that get executed many times. Code that is not executed frequently can be written using conventional methods. Also, a version of the Cray vectorizing compiler is available for IRAM, which allows coding in C, C++, and Fortran. We envision three levels of coding for IRAM. At the highest level, conventional C, C++, or Fortran will be used. For inner loops and frequently called subroutines, the source can be annotated (with `#pragmas`) to vectorize more efficiently. Finally, for the most important computational kernels, highly efficient hand-optimized library routines will be provided. Several such kernels, including the fast Fourier transform [8], motion estimation, and various linear algebra operations are already available.

The primary disadvantage of VIRAM-1 from the speech recognition viewpoint is the limited RAM. This is not a limitation of the architecture, but rather of the current implementation. IRAM can access off-chip memory, but it is unclear at this time how much, if any, will be available.

2.4. Status

VIRAM-1 is in the final stages of design. Tape-out is scheduled for summer of 2001. Algorithm development and testing is conducted on simulators, both at the instruction set level, and a detailed, clock level performance simulation. The simulators, especially the performance simulator, operate fairly slowly, and are therefore mostly useful for testing and analyzing computational kernels, rather than entire programs.

3. AUTOMATIC SPEECH RECOGNITION

For SpeechCorder to operate as a portable unit, it is necessary for the speech recognition algorithms to run efficiently on IRAM. To this end, we have begun to implement a speech recognition system with vectorized algorithms. We based the system on our existing desktop speech recognition algorithms.

Figure 3 shows a block diagram of ICSI's hybrid speech recognition system [4]. Sounds are digitized from a microphone, and are delivered to the Signal Processing unit, typically at 16,000 values per second and 16 bits per value.

3.1. Signal Processing

The Signal Processing unit performs feature extraction, in which the linear amplitude signal is converted to a sparser, spectral-like representation.

Generally, vectorizing signal processing is fairly easy, as the computational bottleneck is usually a filterbank, followed by processing on independent channels. A filterbank can be implemented as a Fast Fourier Transform (FFT). Vectorization of the FFT has been extensively studied, and an efficient implementation that has been validated against the simulators is available for IRAM [8].

3.2. Phone Probability Estimation

The features produced by the Signal Processor are passed to the Phone Probability estimator. This component estimates the probability that the given time interval represents a particular sound in the language (for example, was the sound a /k/, or an /a/, or a /p/). For each sound in the language (typically between 48 and 64 in English), the Phone Probability Estimator outputs a value between 0 and 1, once every time interval. In some systems, a larger number of sound units are used to represent the effects of context and variation within a phone.

At ICSI, we implement the Phone Probability Estimator using a multi-layer perceptron (MLP) neural network. A somewhat more common approach incorporates gaussian mixtures, but other groups have successfully used recurrent neural networks, decision trees, and support vector machines.

An MLP is easy to vectorize. The operations consist of multiplying an input vector by a weight matrix, and then applying a non-linearity to each element of the resulting vector. For efficiency, several input vectors are usually queued up, and a matrix-matrix multiply (rather than a matrix-vector multiply) is performed. This is the computational bottleneck of the Phone Probability Estimator. Note that matrix-matrix multiply can be performed at high speed on vector architectures [5], and an implementation that achieves more than 90% of peak performance on the simulator is available for IRAM.

3.3. Decoding

Conceptually, the decoder takes the sequences of estimates of the phone probabilities, and compares them against models of every permissible word sequence. It then outputs the most likely utterance. For unrestricted large vocabularies

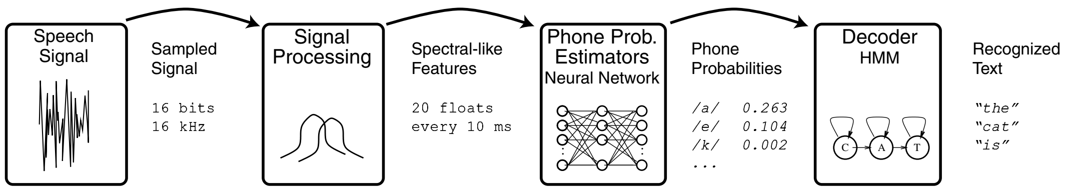


Figure 3: ICSI's hybrid speech recognition system.

as might be used in natural meetings, however, the search space must be massively pruned for the process to be computationally tractable.

The decoder is implemented as a Hidden Markov Model (HMM) of concatenated sub-word units (e.g. phones), with one HMM per word in the vocabulary. The sequence of words is computed by combining the probability of each individual word according to an HMM with the language model, which provides a score for a given sequence of words based on the likelihood of the sequence according to some model of how words group in the language. Because of the limited memory on VIRAM-1, it is likely we will use a either a reduced language model, or, if off-chip memory or disk is available, we will use a cached language model.

Once likely sequences of words have been computed, the system must store the data so that indexing, querying, and retrieval can be performed.

The decoder is usually the most computationally and memory intensive component of an ASR system. It is also the most challenging to vectorize. One approach is vectorize across words — N words with identical length are computed simultaneously, where N is the vector length. This approach has the advantage of being easy to vectorize. However, organizing the search for the best sequence of words becomes very difficult and memory intensive. Since VIRAM-1 is limited to 16 MBytes of memory, we are investigating another algorithm that vectorizes the Viterbi algorithm during the extension phase of a stack decoder [7] (the extension phase consists of checking every possible word starting at a particular frame). This has the advantage of using conventional scalar algorithms for the search portion of the algorithm, and efficient vector code for the inner loop. However, the algorithm is somewhat complex to code. We are in the process of developing this algorithm, and have tested one part on the IRAM simulator.

4. SUMMARY

We are in the process of designing and building SpeechCorder, a portable device that records natural meetings in real-time. Earlier versions of the recognition algorithms showed sufficient accuracy to permit information retrieval on the recorded speech. By using VIRAM-1, a new embedded memory vector processor design, speech algorithms should be able to run efficiently on a small platform with low power consumption. To work effectively on IRAM, speech algorithms must be vectorized. We have implemented a portion of the speech recognizer, and tested it on the IRAM simulators. The result was a performance that was 80% of the peak predicted for the design. We are now in the process of finalizing vectorization of the Decoder. Our IRAM colleagues are nearly done with the chip design, and we expect to be testing the algorithms on the actual chip later in the year. In a separate project, we are working on improving far-field microphone recognition for data collected during meetings at ICSI.

More information on SpeechCorder and IRAM can be found on the web [1] [2].

5. ACKNOWLEDGMENTS

The authors would like to thank the IRAM group at Berkeley, DARPA, NIST, SRI, IBM, and the University of Washington for their participation in the project.

REFERENCES

[1] IRAM web pages. http://iram.cs.berkeley.edu.

[2] Meeting recorder web pages. http://www.icsi.berkeley.edu/real/mtgrcdr.html.

[3] D. Abberley, S. Renals, and G. Cook. Retrieval of broadcast news documents with the THISL system. In *Proceedings IEEE Int'l Conference on Acoustics, Speech, & Signal Processing*, pages 3781–3784, 1998.

[4] H. Bourlard and N. Morgan. *Connectionist Speech Recognition: A Hybrid Approach.* Kluwer Academic Publishers, 1993.

[5] F. G. Gustafson J. J. Dongarra and A. Karp. Implementing linear algebra algorithms for dense matrices on a vector pipeline machine. *SIAM Review*, 26:91–112, 1984.

[6] Christoforos E. Kozyrakis and David Patterson. A new direction in computer architecture research. *IEEE Computer*, November 1998.

[7] D. Paul. An efficient A* stack decoder algorithm for continuous speech recognition with a stochastic language model. In *Proceedings IEEE Int'l Conference on Acoustics, Speech, & Signal Processing (ICASSP-92)*, pages 25–28, San Francisco, 1992.

[8] R. Thomas and K. Yelick. Efficient FFTs on IRAM. In *Proceedings of the 1st Workshop on Media Processors and DSPs (MICRO-32)*, Haifa, Israel, November 1999.