

# ASR System Comparison on New High Performance Workstations \*

G.Antoniol, F.Brugnara, M.Cettolo, R.Fiutem, R.Flor and G.Lazzari

Istituto per la Ricerca Scientifica e Tecnologica (IRST),  
38050 Povo di Trento, Italy

## Abstract

In the last few years the computational power of general-purpose machines has dramatically increased; furthermore high performance workstations quite often offer multimedial capabilities. Therefore it is now possible to implement automatic Speech Recognizers (SRs) without dedicated hardware devices such as DSP boards.

In this paper performance and costs for different configurations of some SRs based on workstations and mixed architectures (workstations equipped with DSP boards), developed at IRST, are presented and discussed. Although the response time depends both on the speech modalities, ( isolated words or continuous speech ), and the hardware platform on which the SR runs, nevertheless, for isolated words, considering the overall system price, a 486 machine can give very interesting results.

## 1 Introduction

Integer and floating point performance of new machines based on the Hp PA-RISC and the DEC Alpha chips can be compared to mainframe computer capabilities. Furthermore, high performance workstations usually offer multimedial capabilities; for instance, the Sun SparcStation 10 and the Hp 735 incorporate a high quality speech input-output device. The general cost decreasing trend for workstation systems will probably make the development of SRs on specialized hardware no longer convenient.

There are already some working prototypes for dictation of Wall Street Journal texts, developed by BBN, CMU [1] and SRI [7]. These prototypes support speaker-independent, real-time, continuous speech dictation, with a vocabulary of 20K words, on a Hp 735. Here for real-time we mean a system that needs a second to recognize a second of speech.

For customer applications of ASR, the key points for success still remain a real-time response and a high recognition rate together with an overall system cost proportional to the actual application benefits.

In the research area, on the contrary, real-time and cost requirements are no longer primary. Actually, in this field it is more convenient to test new algorithms in batch mode

---

\*The work presented here is part of Eureka-FIRST (Friendly Interactive Robot for Service Tasks) project

rather than develop complex real-time code. Performance and costs for different configurations of some SRs developed at IRST are presented and discussed in this paper. The configurations include systems based on workstations and mixed architectures (workstations equipped with DSP boards). Among these there are speaker-independent, medium vocabulary, continuous speech recognizers, and speaker-dependent, large vocabulary, isolated word SRs.

Possible conclusions are that, although the same Hidden Markov Model (HMM) based SR gives very different response times on a Hp 735 with respect to a 486 machine, nevertheless a voice activated menu running on a 486 machine with a low cost A/D board can give a *reasonable* response time at a very low cost. The reported times show that large SRs assuring *real-time* response can be developed and implemented on the new workstations.

This paper is organized as follows. In Section 2 various ASR hardware architectures adopted at IRST are described. Section 3 discusses software issues with respect to hardware architectures and real-time requirements. Section 4 concentrates on performance and cost evaluation of the systems previously described for different application domains. Section 5 draws some conclusions about the evolution of ASR system architectures in the next few years.

## 2 SR hardware architectures

Three different hardware configurations for the implementation of an SR have been considered:

1. an integrated AD/DA performing signal acquisition and all the remaining processing in charge of the workstation processor;
2. like the previous one, but with an external AD/DA board (in the case of a PC for example);
3. a mixed solution: a DSP board for signal acquisition and parameter extraction and the recognition algorithms performed via software;

The various configuration features are briefly summarized in Table 1.

	AD/DA	Parameter Extraction	Pattern Matching
1	Internal	Software	Software
2	AD/DA Board	Software	Software
3	DSP Board	DSP Board	Software

Table 1: Configuration features.

The first configuration has the advantages of being almost completely portable on different hardwares (only the AD/DA driver has to be reimplemented). Furthermore, workstations offer good development environments, they are open to possible hardware enhancements, architectural modifications are not problematic and finally there is no further cost

for the AD/DA. However, this solution has a drawback: handling the AD/DA directly by the operating system may cause loss of data. In fact, assuming a sampling rate of the input signal of 16kHz, an interrupt is generated by the AD/DA every 62.5  $\mu$ s, so the AD/DA driver must take over the CPU control to correctly process the input sample. In the next section we will give more details about this problem.

The second configuration is an obliged choice for the PC machines, that, in almost all cases, have no incorporated AD/DA boards; anyway, today the cost of AD/DA boards is not prohibitive.

The power of today's PC makes this configuration suitable to SRs with a small vocabulary (such as menu driven systems for example) in which real-time recognition is possible.

The third configuration has several disadvantages. First of all, the cost of a DSP board is not negligible. Moreover, the development environments on DSP boards are usually poor, making modifications to the DSP software costly.

The main advantage of this configuration is that the CPU is freed of a remarkable amount of number crunching, that is performed by the DSP. The DSP and the CPU can proceed in pipeline, thus enhancing the system performance. However, a full exploitation of the DSP power can be obtained only at the price of a heavy optimization of the DSP code, thus requiring direct assembler programming and deep understanding of the underlying DSP architecture. This means that porting an application onto another DSP platform requires excessive work.

Table 2 summarizes the comparison of the configurations in terms of portability, development environment, AD/DA driver constraints and cost.

	1	2	3
Portability	High	High	Medium
Development Environment	Good	Good	Medium
AD/DA driver Constraints	High	High	Low
Cost	High	Low	High

Table 2: Comparison of the configuration.

### 3 Real-time software issues in SR development

In this section the software implications of the different configurations previously presented are analyzed.

We start by discussing issues regarding the first and second configurations, in which all the processing, starting from the AD/DA handling, is made via software by the CPU.

We can identify two critical components:

- **input signal acquisition and parameter extraction:**

here the loss of data may compromise all the remaining processing, thus the AD/DA

driver is fundamental;

- **pattern matching:**

the speed is not critical as in the previous point, although it lowers the response time of the whole system.

Following, the signal acquisition and parameter extraction will be referred to as Signal Processing (SP). To handle these components, three different policies are possible, depending on the operating system:

1. the AD/DA driver has a higher priority in the system: this is obviously possible in a real-time operating system, where the response time to asynchronous events can be guaranteed within a predictable time, or in other operating systems (for example Hp Unix) where it is possible to set real-time priority of execution for a process (using the *rtprio* command);
2. the operating system kernel has enough resources to ensure a correct handling of the AD/DA: anyway this requires a careful tuning of the AD/DA driver;
3. in a *multithreaded* system the AD/DA could be handled by an ever active thread; then a recognition thread could process the output data.

Anyway, a common denominator for these policies is the fact that the AD/DA driver must embed an end-point detector that extracts only the parameters corresponding to the meaningful signal; this to avoid an unnecessary overhead for the whole system.

Even though the configurations based on a DSP board do not suffer from the problems outlined before, a driver for the board, although not too complex, is still required. In this case the board can temporarily store the input samples in its memory.

## 4 ASR system comparison

The pattern matching algorithm in all our applications is based on HMM. The figures presented in this paper are obtained with discrete HMMs since they are less computationally expensive than continuous or semicontinuous HMM. A simple left to right topology is used to represent a set of context independent italian phonemes and phonemes in contexts that have been used to build word models [2].

For the continuous speech recognizers the set of legal sentences, commonly referred to as the language model (LM), is expressed with bi-grams [5] represented with an automaton which is used by the Viterbi algorithm [10] [9] in the pattern matching phase. For isolated word SRs the search is constrained to the legal set of words given the last recognized word according to the LM.

### 4.1 Task definition

All the architectures presented are pipeline systems, in which as soon as a new speech signal is detected, an algorithm starts extracting parameters for the ASR subsystem. However,

depending on the different hardware configurations and/or computational power, the recognition process can be delayed until the end of the input signal is detected or can be started immediately. Hence system performance is not only related to the SR but also to the degree of pipelining between the components of the system.

It is well known that *MEL* [6] parameters together with their derivatives, the energy and the energy derivative, are commonly used in the speech community. Discrete HMM have been used in all the architectures to guarantee the fastest SR implementation; therefore the parameter extraction data reported in this paper comprise both parameter extraction and vector quantizations.

In the evaluation of SRs, we have considered continuous speech and isolated word applications.

As application examples to test we chose two tasks related with two projects that are under development at IRST, the Eureka First and the ARES project. In the Eureka First project [4] the application scenario is that of a remote operator who interacts with the navigation system of a mobile robot. In particular, the operator gives navigation commands (e.g. "follow the corridor until the next cross"<sup>†</sup>) and monitoring commands (e.g. "show me the view from the on board camera"). The spoken language interface is supposed to operate in real-time on a workstation, with continuous speech and speaker independent modality, and with flexibility with respect to the accepted language.

The ARES project (Automatic REporting by Speech) project is being developed at IRST in collaboration with the radiologic department of S.Chiera Hospital, Trento [3]. The purpose of the project is the implementation of an interface for dictating, recording and printing radiological reports. At present, the system allows free dictation of reports related to chest and ultra-sound examinations with a small pause in between words. The vocabulary used by the recognizer is medium-sized (2500 - 5000 words) and it is integrated with a statistical language model; run-time modification of the vocabulary and language model is possible, through the insertion of new words and new rules. Table 3 summarizes the main features of the two tasks.

	ARES	Eureka-First
Units' number	52	36
Dictionary size	2500	200
Perplexity	1000	20
Speaker independent	No	Yes
Speech modality	Isolated words	Continuous

Table 3: Task characteristics.

## 4.2 Results

The hardware platforms on which the chosen tasks have been tested are listed in Table 4, together with their respective prices.

---

<sup>†</sup>The actual employed language is Italian, English translations are given for clarity.

Table 4 shows the computation time signal duration ratio for signal processing ( column 3 ) and pattern matching ( columns 4 and 5). The 486 implementation is an architecture of the second type (see Section 2 while all the remaining implementations are of the first type. The rather surprising data is that even a 486 at 66 MHz has enough computational power to process speech signals in real-time but unfortunately this requires about 60% of the CPU power. This means that, even if *fully* pipelined, a complex ASR system will not run in real-time on a 486.

Machine	Price in dollars	SP	ARES	FIRST
486	3000	0.563	1.40	5.9
Sparc 1	10000	1.152	2.24	8.27
SparcClassic	5000	0.45	1.13	4.2
Sparc 10	30000	0.184	0.70	2.67
Hp 730	26000	0.131	0.51	1.62
Hp 735	33000	0.067	0.30	1.25

Table 4: Comparison of Cpu Requirements for ARES and FIRST tasks: Signal Processing (SP) phase is common. The machine prices are for Italian customers.

Table 4 also shows that while for the ARES task, starting from the Sparc10 up to the Hp735 we can reach real-time processing, for the FIRST task, none of the examined architectures has shown real-time performance, with the standard Viterbi search. Considering that the total recognition time comprises also the SP phase (third column) and given the used algorithms, there is no way to have real-time performance for FIRST.

However the real-time requirement can be achieved with a beam search [8] strategy at the cost of sometimes losing the solution. Adding a DSP board to the system increases the cost but does not allow real-time since signal processing is only a small fraction of the overall recognition time.

To summarize and combine the data reported in Tables 4 we introduce a parameter called  $\rho$ , representing the relative percentage of "real-time" per dollar spent for a certain task and hardware platform. Actually,  $\rho$  is a sort of performance cost ratio; it is defined as:

$$\rho = S/C$$

where  $C$  is the platform cost and  $S$  is the relative speed for the different hardware referred to real-time.  $S$  is defined as the inverse of the overall computational cost  $T$ , that is the sum of the signal processing cost ( $SP$ ) and the pattern matching cost ( $PM$ ) (third column plus fourth column of Table 4).

$$S = 1/T = 1/(SP + PM)$$

Figure 1 gives the values of  $\rho$  for each hardware platform considered, using the computational times of the ARES task. An Hp 735 gives good performance for this task, but at a high cost. Anyway, what is worth noticing is that, for an isolated words task like ARES, low cost machines like a 486 PC or a SparcClassic give the highest values of  $\rho$ . Moreover, decreasing LM perplexity decreases system requirements, and, starting from 100

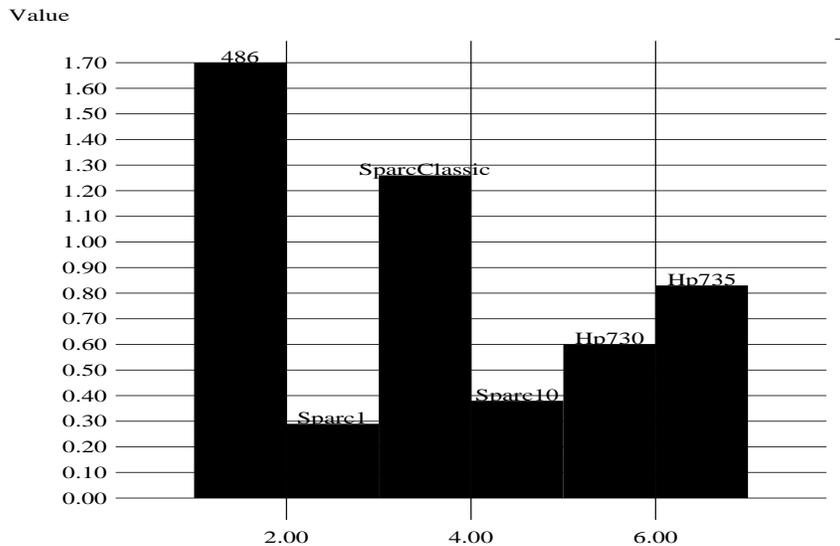


Figure 1:  $\rho$  values (multiplied by 10000) on ARES task.

words perplexity, a 486 system is almost real-time (pattern matching reduced from 1.40 to 0.93). Although a similar diagram can be drawn for the FIRST task, we have to notice that, while for isolated words low cost machines are near real-time, for continuous speech they are about 5 time slower than real-time.

## 5 Conclusions

In the evaluation of SRs two tasks have been used: continuous speech recognition and isolated words recognition.

The application areas are sophisticated recognition/understanding systems for the first kind and isolated word dictation or voice driven menu systems in the second one. The result of our tests show that even on small and not expensive machines we can have very satisfactory results, especially on isolated words.

Better performance for all the hardware platforms considered can be achieved with a beam search algorithm that prunes the search space at the cost of losing, sometimes, the right solution.

Future work will be concentrated in this direction, in fact, a good compromise between recognition rate and speed involves, especially on low cost machines, the use of pruning strategies.

## References

- [1] F. Alleva, X. Huang, and M.Y. Hwang. An improved search algorithm using incremental knowledge for continuous speech recognition. In *Proceedings of the IEEE International*

*Conference on Acoustics, Speech and Signal Processing*, pages II-307-310, Minneapolis, Mn, USA, 1993.

- [2] Bianca Angelini, Fabio Brugnara, Daniele Falavigna, Diego Giuliani, Roberto Gretter, and Maurizio Omologo. A baseline of a speaker independent continuous speech recognizer for the Italian language. In *Proceedings of the 3rd European Conference on Speech Communication and Technology*, Berlin, Germany, September 1993.
- [3] G. Antoniol, F. Brugnara, F. Dalla Palma, G. Lazzari, and E. Moser. A.R.E.S.: An interface for automatic reporting by speech. In *Proceedings of the European Conference on Speech Communication and Technology*, Genova, Italy, 1991.
- [4] G. Antoniol, M. Cettolo, and M. Federico. Techniques for robust recognition in restricted domains. In *Proceedings of the European Conference on Speech Communication and Technology*, Berlin, Germany, 1993.
- [5] L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):179-190, 1983.
- [6] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357-366, 1980.
- [7] H. Murveit, J. Butzberger, V. Digalakis, and M. Weintraub. Large-vocabulary dictation using SRI's DECIPHER<sup>TM</sup> speech recognition system: progressive search techniques. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages II-319-322, Minneapolis, Mn, USA, 1993.
- [8] N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga: Palo Alto, CA, 1980.
- [9] L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, pages 4-16, Jan 1988.
- [10] A. J. Viterbi. Error bounds for convolutional codes and asymptotically optimum decoding algorithm. *IEEE Trans. on Information Theory*, 13(2):260-269, 1967.