# WORD RECOGNITION USING ERROR-CORRECTION CODES

Leisa Condie,

Department of Computer Science,
University College, University of New South Wales.

The Reed-Solomon error-correction code separates input vectors as far as possible from each other. Such codes are known as Maximum Distance Separable (MDS). This property was investigated in a word recognition system to see whether applying such a code would separate word vectors to such a point that recognition rates improved. A vocabulary of 21 words spoken on four occasions by a single speaker formed the basis of the experiment. First formants for each frame were found with the Interactive Laboratory System (ILS) package. The resulting vectors were encoded with a Reed-Solomon code. The reference set for recognition was formed from the average of all the utterances of each word, and a simple distance metric (after suitable Dynamic Time Warping to align the vector lengths) used to find the closest reference word. A comparison of performance for encoded and unencoded vectors is made.

## INTRODUCTION

The aim of error-correcting codes, particularly those which are Maximum Distance Separable, is to scatter input vectors such that two vectors cannot be confused even if distorted by noise in the transmission channel. Codes can be designed to match the expected numbers of errors and, so long as that number of errors is not exceeded, the original vector can be restored after distortion. McGuinness (1987) found that encoding vectors composed of spectral amplitudes for a vocabulary of ten words spoken by ten people resulted in all the vectors scattering. It was the aim of this investigation to extend this experiment to find whether vectors of the same word spoken by the same person would cluster, whilst vectors for other words would separate, thus improving recognition.

## ERROR-CORRECTION CODES

Error-correction codes operate over Galois Fields of $n$ elements, denoted by GF(n), from zero to $n-1$. All arithmetic operations are performed in this field: i.e. all operations are performed modulo $n$. A Reed-Solomon code is an MDS code which can be designed for various error rates and which deals with an arbitrary alphabet. These features made this code ideal for these experiments.

The generator polynomial for a Reed-Solomon code is given by

$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3)...(x - \alpha^{\delta - 1})$$

where $\alpha$ is a primitive root of GF(n): i.e. $\alpha$ is a number such that when $\alpha$ is raised to the powers $1, 2, ...n-1$ (taken modulo $n$), the values do not repeat, and also map every element of GF(n). The designed distance of the code, $\delta$, is defined by $\delta = 2 \times$ number of errors correctable + 1. The number of errors depends on an analysis of the channel for noise which is not a concern here. As the length of the input vector has to equal the number of rows in the generator matrix, the number of errors was chosen in such a way as to make the matrix match, as closely as possible, the input vector length.

An example: for GF(41), $\alpha = 7$ and $\delta = 15$, the generator polynomial is

$$g(x) = x^{14}+11x^{13}+24x^{12}+31x^{11}+20x^{10}+28x^9+34x^8+x^7+x^6+28x^5+26x^4+38x^3+11x^2+39x+35$$

This is used to form a matrix of dimensions $(n-\delta) \times (n-1)$, which in this case gives $26 \times 40$. Each input vector is multiplied by this generator matrix (in GF(n)) to give the encoded vector. The generator matrix has the form:

[35 39 11 38 26 28 1 1 34 28 20 31 24 11 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[ 0 35 39 11 38 26 28 1 1 34 28 20 31 24 11 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
. . .

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 35 39 11 38 26 28 1 1 34 28 20 31 24 11 1]

METHOD

Four sets of a vocabulary of 21 words were digitised at 8kHz and the endpoints of each word determined by hand. The 21 words included one and two syllable words as well as similar words. The ILS package was used to extract the first three formants and bandwidths of each frame in each word by root solving. A preliminary investigation was made in which a single set of three formants and three bandwidths, the average over the entire word, was used, whilst a second experiment used vectors formed from the first formant for each frame in the word.

In both methods vectors were mapped onto GF(n) by rounding to the nearest 50 and dividing by 50. This method was found to best preserve the shape of the vectors, eliminating distortion as a possible source of poor performance. The value of $n$ used depended on the range of values the vector elements encompassed and could be easily changed. The following graph shows an original vector (dotted line) and the mapped vector in GF(41) (solid line) before the final division. It is clear that this mapping retains the shape of the formants.

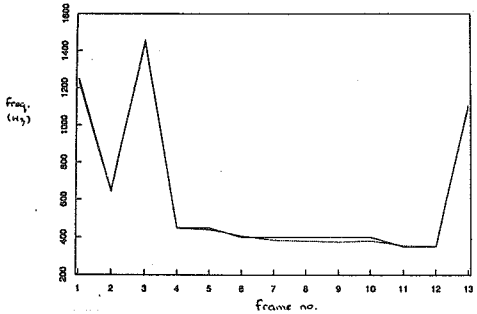


Figure 1. Word "two" before mapping (dotted) and after (solid).

Two experiments were performed on the average vectors, the first involving only a handful of words, the second the whole vocabulary. The aim of each was to eliminate a possible sequence of operations for forming the reference set so only the optimal would be performed in the slower second method. Examination of the results for these experiments show that forming the reference set from test vectors that had been encoded without mapping gives poor results. It was also found that references formed by averaging after mapping into GF(n) were superior to those formed by mapping before averaging. Further, encoding the reference vectors should be done after forming the average vector for the word, rather than before. In this way, the basis for the second experiment was laid. In both experiments the number of errors corrected by the code used was chosen so that the input vector length was 8. As the average vector was of length 6 two zeroes were added to the end as padding and the resulting zeroes in the output discarded.

The second experiment involved vectors of first formants. The first stage was to map onto GF(41). Simple interpolation was used to make all vectors the same length. The length was chosen on the basis of the samples for each word. This was followed by dynamic time warping: the normalize/warp method of Myers, Rabiner and Rosenberg (1980) was used. For each word warping gave vectors of the same length - or else it was possible to divert through regions of zero difference to obtain the same length - and these were averaged to form the reference set. The code chosen took vectors of length 26 so all test and reference vectors were interpolated to this size (except for two reference vectors) and encoded. These were warped against each encoded reference and a provisional score recorded in the

Results section.

RESULTS

In all of the results below the reference set of vectors was formed by averaging the test vectors for each word then comparing all of the test vectors against them with a simple Euclidean distance metric. If the test vectors were raw then the reference was formed from the raw. Similarly if the test vectors were encoded mappings, the reference vectors were averages of these. The first method tried involved an average over the whole word. The first experiment involved only two utterances each of five test words.

| Reference | Recognition |
|---|---|
| Raw | 100% |
| Raw encoded | 20% |
| Mapped | 90%† |
| Mapped then encoded | 60% |

Table 1. Results for method one, experiment one.

"Raw" indicates the test vectors were not treated in any way whilst "Raw encoded" indicates the test vectors were encoded in GF(23) with a 7-error-correcting code. † indicates there was a test returning two possible answers which was not included in this figure. "Mapped" means the test vectors were mapped onto GF(23) as outlined in the Methods section, whilst "Mapped then encoded" indicates that encoding took place after mapping. Encoding the raw vectors was clearly detrimental and was no longer considered.

The second experiment with the average vectors involved four sets of the full 21 word vocabulary. Although not shown in the table below, a comparison was made of recognition when the reference was formed by averaging the test vectors before the mapping occurred and after. It was found that averaging after the mapping was far superior in performance.

| Reference | Recognition |
|---|---|
| Raw | 82% |
| Mapped | 73%† |
| Mapped, averaged then encoded | 20% |
| Mapped, encoded then averaged | 10% |

Table 2. Results for method one, experiment two.

In this case the mapping is onto GF(31) and the code is 11-error-correcting. The comparison of the effect of averaging before encoding with averaging after clearly shows averaging before to be superior, but neither recognition rate is impressive. † indicates an actual recognition rate of 54% plus an additional 19% which returned two answers. An additional experiment was performed with the mapped vectors where a maximum was placed on element-to-element distances to see the effect on performance. With a maximum of 2 the recognition rate was 65% (including 14% ambiguity) whilst a maximum of 1 gives a recognition rate of 65% (including 13% ambiguity).

The second experiment involved vectors formed from first formants only. Although all of the words were tested, two were left out of the reference set. In the following table the mapping was in GF(41) and the code was 7-error-correcting. † indicates an actual recognition rate of 69% plus an additional 1% which returned an ambiguous answer. At the time of writing the recognition rate for the encoded was provisional (only 1/4 of the tests had finished).

| Reference | Recognition |
|---|---|
| Mapped | 70%† |
| Mapped, averaged, encoded | 8% |

Table 3. Results for experiment two.

## CONCLUSION

The idea of using error-correction codes to improve recognition rates is invalid due to the object of error correction being to force vectors, particularly those which are close and hence confusable, apart. Whilst this is desirable in preventing noise from confusing two vectors, it is a hindrance when it is the aim to cluster close vectors, and further separate more distant ones. If vectors for a given word were always the same it would work, but this is of course completely unrealistic.

It was also found that the order of performance of operations on speech vectors can have a significant effect on the performance of the recogniser and no matter what techniques are in use, this factor must be considered when performing experiments.

## REFERENCES


Condie, L. (1987) *Noise Removal in Isolated Word Recognition Using Error-Correction Codes*, Honours thesis, Basser Dept. of Computer Science, University of Sydney.

MacWilliams, F.J. & Sloane, N.J.A. (1977) *The Theory of Error-Correcting Codes*, (North-Holland Publishing Company: New Jersey).

McGuinness, H. (1987) *Speech Recognition Using Error Correcting Codes*, Honours thesis, Basser Dept. of Computer Science, University of Sydney.

Myers, C., Rabiner, L.R. & Rosenberg, A.E. (1980) *Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition*, IEEE Trans. on Acous., Speech and Signal Processing, ASSP-28, 6, 623-635.

# A SIMPLE PITCH DETECTOR USING A DIGITAL SIGNAL PROCESSOR

R.E.E.Robinson
Speech, Hearing, and Language Research Centre
Macquarie University

ABSTRACT - A Real Time Pitch extraction device using the TMS32010 Digital Signal Processor is described. It is designed as a replacement for an analog Pitch extractor and performs the same function but with greater accuracy and better dynamic response. The two are compared.

## INTRODUCTION

The Macquarie University has a Real Time Pitch Meter which has been performing well for a number of years. The Pitch Meter was designed by the Speech, Hearing, and Language Research Centre (SHLRC) and consists of analog circuitry. The data from it is fed to a computer where it is printed or saved to disc. The computer merely logs the data. The analog Pitch Meter is built in a modular fashion which lends itself to easy redesign and improvement as such is needed in the research field. In this case the Pitch Measurement module has been replaced by another module using a Digital Signal Processor (DSP).

## EXISTING PITCH METER

The Pitch Meter consists of an analog Pitch processor and a computer for data logging. The analog circuitry consists of a Microphone Amplifier for live analysis, and a Line Amplifier for connection to a tape recorder for analysis of recorded speech. (see Figure 1) A Level Meter is provided to allow levels to be set up correctly. A Low Pass Filter (LPF) is used to allow only the low frequencies through, which contain the pitch information. The Filter is manually switchable to different ranges to allow for different pitch voices and to overcome second harmonic influences. The signal is then split to an Intensity measuring module and the Pitch measuring module. The Intensity module gives an analog output proportional to the signal intensity. This is displayed on the computer as an Intensity envelope and is used to distinguish voicing from nonvoicing. The Pitch module gives an analog output proportional to the period of the incoming signal. This is interpreted as the Pitch and is displayed on the computer, time aligned with the Intensity envelope. This occurs in real time.

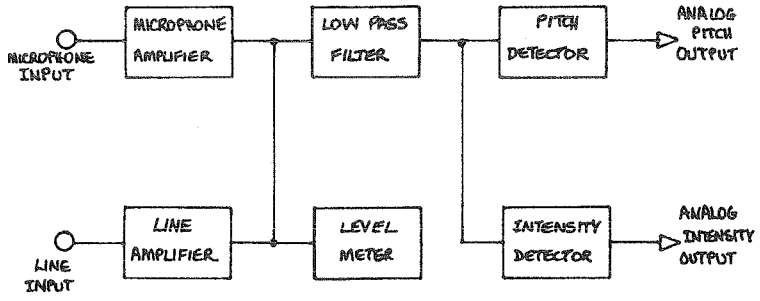


*Figure 1. Pitch Meter Block Diagram*

EXISTING ANALOG PITCH DETECTOR

The existing Pitch detector is an analog device and functions very well, however there is room for improvement. It is based on the VFQ1 voltage to frequency converter Integrated Circuit (IC) by DATEL. The input to the module is buffered and a comparator acts as a zero crossing detector (squarer) generating a square wave output at the applied frequency. (see Figure 2) The square wave then goes to the VFQ1. This IC is connected as a frequency to voltage converter. An analog output is generated proportional to the input frequency. This technique tracks the Pitch quite well but suffers from capacitor charge and discharge delays. The output from the Pitch detector is typically 30 milliseconds (mS) behind the input, as it takes this length of time for the charging waveform to allow it to reach the correct voltage output. When voicing stops, the capacitor then discharges, and this takes about 25mS. The output of the Pitch detector has some ripple and is smoothed by a 25 Hertz (Hz) LPF. Despite these shortcomings the Pitch detector works well under most conditions. To overcome these problems, a new one was designed, taking advantage of DSP technology.

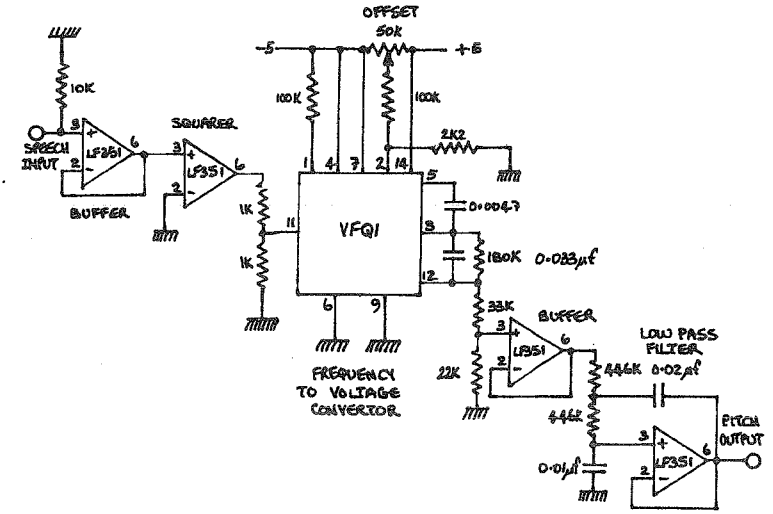


*Figure 2. Analog Pitch Detector Circuit*