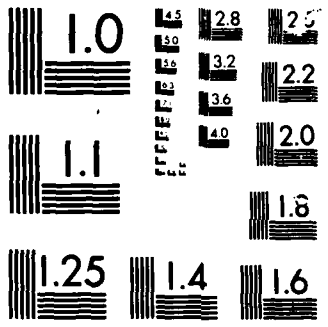
AD-A166 921   ADVANCED ARCHITECTURES FOR DIGITAL SIGNAL PROCESSORS   1/2
(U) MITRE CORP BEDFORD MA A L BEQUILLARD ET AL. OCT 85
MTR-9647 RADC-TR-85-203 F19628-84-C-0001

UNCLASSIFIED   F/G 9/2   NL



MICROCOP' CHART

(2)

RADC-TR-85-203
Final Technical Report
October 1985

AD-A166 921

ADVANCED ARCHITECTURES FOR DIGITAL SIGNAL PROCESSORS

The MITRE Corporation

A. L. Bequillard, D. O. Carhoun and W. L. Eastman

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC
S ELECTE D
MAY 0 6 1986
E

DTIC FILE COPY

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

86 5 5 073

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-85-203 has been reviewed and is approved for publication.

APPROVED: Stanley Lis

STANLEY LIS
Project Engineer

APPROVED: [illegible]

FRANK J. REHM
Technical Director
Surveillance Division

FOR THE COMMANDER: John A. Ritz

JOHN A. RITZ
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (OCTS) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A166 921

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) MTR 9647	5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-85-203
6a. NAME OF PERFORMING ORGANIZATION MITRE Corporation; 6b. OFFICE SYMBOL (If applicable) D-82	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (OCTS)
6c. ADDRESS (City, State, and ZIP Code) Burlington Road Bedford MA 01730	7b. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center; 8b. OFFICE SYMBOL (If applicable) OCTS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19628-84-C-0001
8c. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
62702F	MOIE	74	40

11. TITLE (Include Security Classification)
ADVANCED ARCHITECTURES FOR DIGITAL SIGNAL PROCESSORS

12. PERSONAL AUTHOR(S)
A. L. Bequillard, D. O. Carhoun, W. L. Eastman

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM Oct 83 TO Oct 84	October 1985	178

16. SUPPLEMENTARY NOTATION
N/A

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
09	03	
17	09	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Signal Processing; Residue Number System(RNS); Speech Recognition; Autoregressive Spectral Estimation; Systolic Architecture; Dynamic Time-Warping

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
This report is concerned with the development and application of improved techniques for digital signal processing, based on use of residue number system (RNS) to implement the processing functions associated with isolated-word speech recognition. Specifically, the use of RNS in combination and systolic architectures for implementation of speech recognition algorithms was explored. The implementation of time-warping speech algorithm in RNS is described.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT] ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Stanley Lis	22b. TELEPHONE (Include Area Code) (315) 330-4437; 22c. OFFICE SYMBOL RADC (OCTS)

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

ABSTRACT

This report is concerned with the development and application of improved techniques of digital signal processing, based on the use of residue number systems (RNS), to implement the processing functions associated with isolated-word speech recognition. It constitutes final documentation, for fiscal year 1984, on MITRE Mission Oriented Investigation and Experimentation (MOIE) project 7440: Advanced Architectures for Signal Processors.

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

QUALITY INSPECTED 3

TABLE OF CONTENTS

Section			Page
LIST OF ILLUSTRATIONS			viii
LIST OF TABLES			xi
1	INTRODUCTION AND SUMMARY		1
	1.1	INTRODUCTION	1
	1.2	SCOPE	2
	1.3	SPEECH RECOGNITION FUNDAMENTALS	3
		1.3.1 Preprocessing of the Speech Waveform	5
		1.3.2 Utterance Detection	6
		1.3.3 Segmentation of the Utterance	9
	1.4	SPECTRAL PROCESSING FOR FEATURE EXTRACTION	11
		1.4.1 Autoregressive Spectral Estimation	13
		1.4.2 The Itakura-Saito Distortion Function	17
	1.5	DYNAMIC TIME-WARPING	20
	1.6	SUMMARY OF RESULTS	20

TABLE OF CONTENTS (Continued)

Section		Page
2	USE OF THE ITAKURA-SAITO DISTORTION FUNCTION FOR SPEECH RECOGNITION	25
	2.1 INTRODUCTION	25
	2.2 A FREQUENCY-INDEPENDENT FORMULATION	26
	2.2.1 Gain Normalization	38
	2.3 A MATCHED FILTERING INTERPRETATION	42
	2.4 COMPUTATION IN A RESIDUE NUMBER SYSTEM	45
3	RESIDUE NUMBER SYSTEM IMPLEMENTATION OF A DYNAMIC TIME-WARPING BASED SPEECH RECOGNITION SYSTEM	47
	3.1 INTRODUCTION	47
	3.2 DYNAMIC TIME-WARPING ALGORITHM	49
	3.2.1 DTW Path Constraints	51
	3.2.2 DTW Path Computations	58
	3.3 RNS IMPLEMENTATION OF DYNAMIC TIME-WARPING ALGORITHM	63
	3.4 QUANTIZATION IN A RESIDUE NUMBER SYSTEM	72
	3.4.1 Introduction	72
	3.4.2 The Quantization Function t(x)	73
	3.4.3 Calculations	80
	3.4.4 Choice of Moduli	84
	3.4.5 Conclusion	85

TABLE OF CONTENTS (Continued)

Section			Page
	3.5	RNS IMPLEMENTATION OF THE SHORTEST PATH ALGORITHM	86
	3.6	RNS RANGE AND SCALING	89
	3.7	SIMULATION RESULTS FOR IMPLEMENTATION OF DTW ALGORITHM IN RNS	90
	3.7.1	Simulation Test Set	90
	3.7.2	Effect of Input Scaling on Recognition Error Rate	91
	3.7.3	Effect of Distortion Function Scaling on Recognition Error Rate	93
	3.7.4	Effect of Quantization Threshold on Recognition Error Rate	93
	3.7.5	Effect of RNS Range on Recognition Error Rate	97
4		SYSTOLIC ARCHITECTURE FOR RNS IMPLEMENTATION	99
	4.1	AUTOCORRELATION COMPUTATION	99
	4.2	LINEAR SYSTOLIC ARRAY FOR AUTOCORRELATION COMPUTATION	103
	4.2.1	RNS Hardware Concept	110
	4.3	SEMISYSTOLIC DTW ARRAY	115
	4.3.1	Computation of the Array of Distortion Values	116
	4.3.1.1	Quantization of the Distortion Values	123

TABLE OF CONTENTS (Concluded)

Section		Page
4.3.2	Shortest Path Computations	126
4.3.3	Interconnection Concepts	135
4.3.4	Estimate of Throughput	137
4.4	SUMMARY	139
5	CONCLUSION	141
5.1	SUMMARY	141
5.2	ALTERNATIVES FOR DISTORTION COMPUTATION	143
5.2.1	Itakura Distortion	144
5.2.2	Euclidean Distance Measures	145
5.2.2.1	Log Spectral Deviation	146
5.2.2.2	Direct Autocorrelation Analysis	147
5.3	FOLLOW-ON RECOMMENDATIONS	148
REFERENCES		151
APPENDIX A:	RESIDUE NUMBER SYSTEMS	155

LIST OF ILLUSTRATIONS

Figure		Page
1.1	Speech Recognition via Dynamic Time Warping	4
1.2	Input Speech Preprocessing	7
1.3	Utterance Detection	8
1.4	Segmentation of an Utterance	10
1.5	LPC/DTW Word Recognition System	21
3.1	Dynamic Time-Warping	50
3.2	Unconstrained DTW Grid	52
3.3	Local Path Constraints	55
3.4	Grid for Type 3 Local Constraints	59
3.5	DTW Network	61
3.6	Distribution of Path Metric Differences	65
3.7	Distribution of Local Distortion Values	68
3.8	RNS Implementation of DTW Algorithm	71
3.9	Calculation of $t(x) = \|s_2(x) - s_1(x)\|_{p_1}$	83
3.10	RNS Implementation: Recognition Error Rate vs. Input Scaling	92
3.11	RNS Implementation: Recognition Error Rate vs. Distortion Function Scaling	94
3.12	Recognition Error Rates	95
3.13	RNS Implementation: Recognition Error Rate vs. Quantization Threshold	96
3.14	RNS Implementation: Recognition Error Rate vs. RNS Range	98

LIST OF ILLUSTRATIONS (Continued)

Figure		Page
4.1	Input Segmentation	100
4.2	Autocorrelation Computer	102
4.3	Systolic Autocorrelation Computer	104
4.4	Autocorrelation Computation	105
4.5	Autocorrelation Computation: Final Steps	106
4.6	Autocorrelation Computation: Two Consecutive Input Segments	108
4.7	Processing Element in Autocorrelation Array	109
4.8	Systolic RNS Autocorrelation Computer	111
4.9	Systolic Correlator Cell Mod P	112
4.10	Modulo P Multiplier: Logarithmic Implementation	114
4.11	DTW Input Data Flow	118
4.12	Systolic Computation of Local Distortion	119
4.13	Computational Wavefront	120
4.14	Local Distortion Computation	122
4.15	RNS Quantizer	124
4.16	DTW Array Processing Element	129
4.17	DTW: Systolic Path Computation	130
4.18	Systolic DTW Computation	131
4.19	DTW Mod P Cell Architecture	134

LIST OF ILLUSTRATIONS (Concluded)

Figure		Page
4.20	RNS DTW Cell	136
4.21	Schedule of DTW Cell Operations	138
4.22	Correlator and DTW Array	140
A-1	General Residue Number System Function	157

LIST OF TABLES

Table		Page
3.1	Differences Arising in the Shortest Path Calculations of the Dynamic Time-Warping Algorithm	66
3.2	Histogram of Differences in DTW Shortest Path Calculations: One-Bit Quantization of Gain-Normalized Itakura-Saito Distortion Function Values - Breakpoint = .5	69
3.3	t(x) for 4, 9, 5, 7 Residue Number System	76
3.4	t(x) for 7, 11, 2, 3, 5 Residue Number System	78
3.5	$\|t(1)^{-1} \cdot t(x)\|_{p_1}$ for 7, 11, 2, 3, 5 RNS	79
3.6	Tables for Quantization in 4, 5, 7, 9 RNS	81
3.7	Number of Overflows of First Modulus and Recognition Error Rates for Various RNS2 Choices for DTW Calculations - One-Bit Quantization of Distortion Values - RNS1 = {73, 71, 67, 61, 59}; Threshold = 16	88

SECTION 1

INTRODUCTION AND SUMMARY

1.1 INTRODUCTION

This report is concerned with the development and application of improved techniques of digital signal processing, based on the use of residue number systems (RNS), to implement the processing functions associated with isolated-word speech recognition. It constitutes final documentation, for fiscal year 1984, on MITRE Mission Oriented Investigation and Experimentation (MOIE) project 7440: Advanced Architectures for Signal Processors.

Speech recognition is a computationally intensive application for digital signal processing in which residue number system techniques can play an effective role in reducing the computational burden, or equivalently, in increasing the throughput rate at a fixed computational level. Under this project, we have explored the use of RNS-based computations, in combination with systolic architectures, for the improved implementation of speech recognition algorithms. Our work has focused on a particular type of word recognition algorithm that is based on an autoregressive model of the speech production process and a dynamic programming approach to effecting time registration between test and stored-reference speech patterns. While other approaches to speech recognition are certainly feasible and the subject of active research at many institutions, the approach we have adopted is representative of the results of many years of speech research occurring in a large segment of the speech processing community. Our objective was not to advance the state of speech research, but rather to concentrate on the RNS implementation of a well-understood and commonly used method of speech recognition. We expect that the RNS implementation advantages demonstrated by our work will extend also to the processing functions resulting from improved speech recognition research.

1.2 SCOPE

In the remainder of section 1 we will present a view of the fundamentals of speech recognition processing sufficient for understanding of our RNS implementation work. In section 2 we will present a self-contained derivation of the Itakura-Saito distortion function from a time-domain viewpoint, the distortion computation being a central issue in any speech recognition process. Section 3 is devoted to a complete description of the dynamic time-warping (DTW) algorithm used for time registration between test and reference patterns and its RNS implementation. Of particular concern is a technique of quantization of the distortion values within RNS, necessary to contain the dynamic range while allowing satisfactory discrimination between word matches and mismatches. Section 4 discusses RNS implementation, in a linear systolic array, of the sample autocorrelation function estimate upon which the distortion computations are based. Also developed is the RNS architecture for a two-dimensional systolic array, or computational wavefront processor, in which the distortion function and dynamic programming computations are carried out. Of particular concern is the pipelining of the computations to maintain a high throughput in the processor. Section 5 summarizes conclusions and presents recommendations for further work. Necessary details of residue number systems and their properties are contained in an appendix.

1.3 SPEECH RECOGNITION FUNDAMENTALS

The rudiments of a speech recognition process are pictured in figure 1.1. In this process the input utterance, which is a word to be matched to one in a reference library of stored utterances, is analyzed in short blocks of overlapping segments from which a spectrogram, or time versus frequency plot, of the utterance may be constructed. Segmentation into short blocks allows the process to be viewed as locally stationary, the time variation being accommodated by the sequential processing of overlapping analysis segments. It is assumed that a similar analysis has been performed on the utterances contained in the reference library. In both cases, feature vectors are compared to produce a local measure of distortion between segments of the test utterance, and those of one of the reference utterances.

If there are n segments of the test utterance and m segments of the reference utterance then the local distortions, based on a Euclidean distance metric or something similar, define a two-dimensional grid of $n \times m$ distortion values, the low values corresponding to good matches between analysis segments and the high values corresponding to poor matches. The purpose of the dynamic time-warping algorithm is to effect time registration between the stored and test segments to compensate for local time expansion or contraction of the test utterance with respect to the reference. It is a dynamic programming algorithm which calculates the accumulated weighted distortions for the least-cost path through the grid of distortion values. This score for the comparison of utterances is compared in magnitude with the scores for other pairings to produce a final decision as to which reference utterance provides the best match.

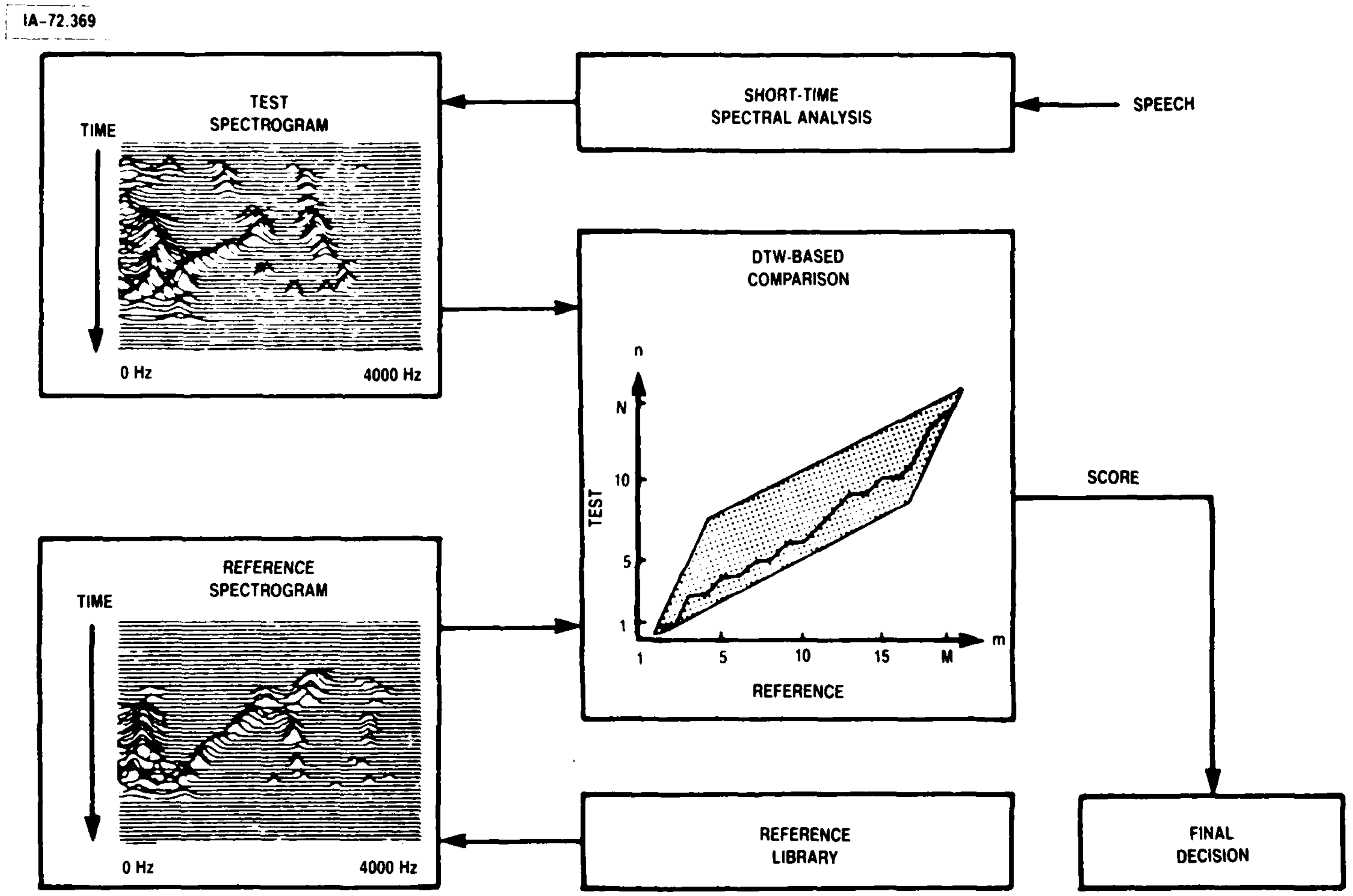


Figure 1-1. SPEECH RECOGNITION VIA DYNAMIC TIME WARPING

The dynamic time-warping computations and distortion function computations impose the greatest computational burden in the recognition process. Although the short-time spectral analysis, or feature extraction, computations can be quite complex, the analysis needs only to be performed once for each segment of the test utterance. Although the distortion and DTW computations may not be as complex, individually, there is a need to produce an array of distortion computations for each pair of utterances coupled with the dynamic programming computations to produce a score comparing the test utterance with each of the reference utterances stored in the library. This is the computational bottleneck in the recognition process that we expect to impact with the combination of RNS computation and systolic array architecture.

1.3.1 Preprocessing of the Speech Waveform

Since the speech recognition process will involve digital computations on overlapping segments of the analog speech waveform, preprocessing of the speech signals is required to obtain the appropriate digital signals. The waveforms must be appropriately sampled and quantized, the beginning and end of an utterance established, the utterance segmented into overlapping blocks, and the segments windowed for subsequent spectral processing. The sampling rate must be high enough to prevent aliasing, the quantization must be sufficient for satisfactory digital representation of the analog signal, and the segment length must be short enough to provide a stationary sample of the spectrum yet long enough to provide adequate spectral resolution. The reference patterns to be stored in the library must be preprocessed in identical fashion to avoid artificial distortion due to processing differences. In our experimental work, we have made use of a computer simulation, operating on the MITRE Corporate Research Computer Facility, which digitally processes speech waveforms that have been preprocessed in our Audio Signal Conversion Laboratory (ASCL).

Input processing of the speech waveforms and their A/D conversion are pictured in figure 1.2. Voice signals are picked up by the microphone, amplified, and passed through low-pass filters to remove frequency components above 4 kHz. After equalization to compensate for a finite sampling aperture, the analog signals are converted to 12-bit digital samples at an 8 kHz sample rate by an A/D converter. Output from the A/D converter is either stored in a designated file for future input to the simulation or, in recognition mode, may be input directly to the speech recognition system implemented in the simulation.

1.3.2 Utterance Detection

Detection of an utterance, as contrasted with a period of silence, is regarded as a digital preprocessing function in our work. In our experimentation, we have based utterance detection on observation of energy statistics. The procedure is pictured in figure 1.3. The energy statistic is a measure of the short-time average signal energy minus the long-time (exponentially averaged) signal energy. Three thresholds are set as shown in the figure. The beginning of an utterance is detected if the energy statistic rises above the START threshold and remains above it until crossing the HIGH threshold. The end of an utterance is detected if the energy statistic falls below the END threshold and remains below it for at least 150 msec. These events constitute a valid utterance detection if the length from beginning to end is at least 240 msec.

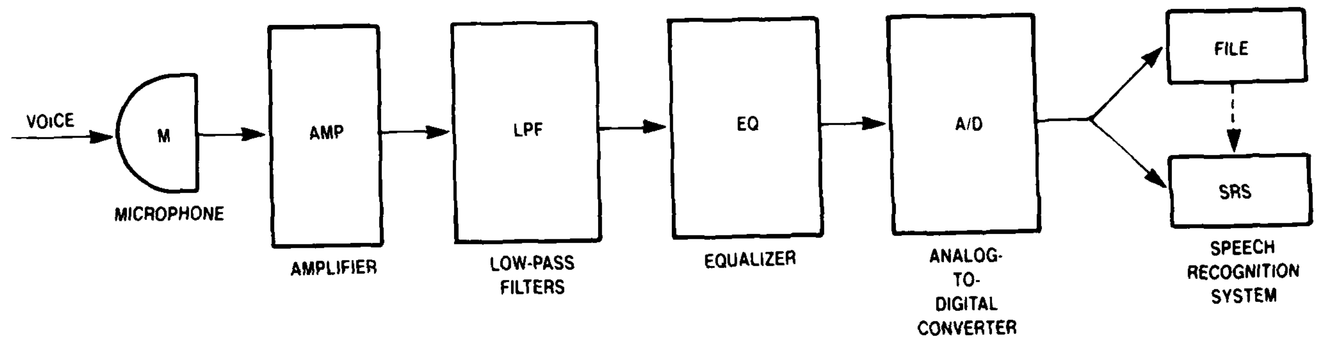


Figure 1-2. INPUT SPEECH PROCESSING

IA-72.368

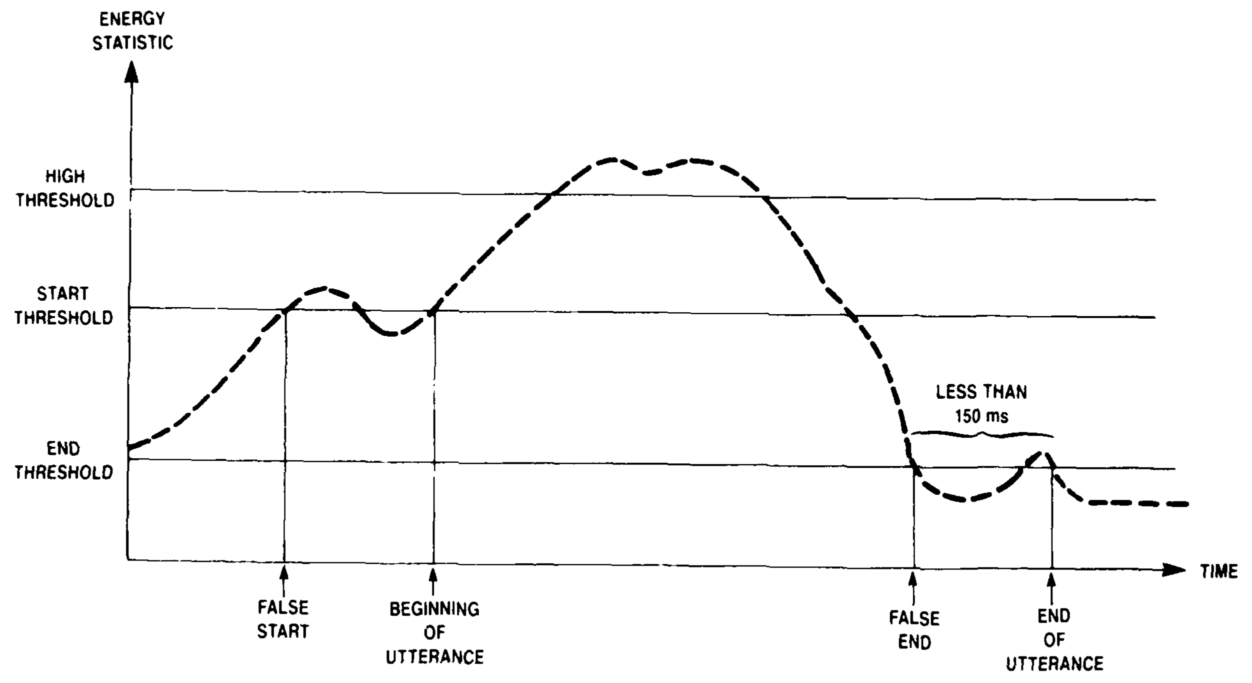


Figure 1-3. UTTERANCE DETECTION

1.3.3 Segmentation of the Utterance

After detection, an utterance must be divided into segments for short-time analysis. The segmentation that we use in our experimentation is shown schematically in figure 1.4. The analysis interval should be long enough for good spectral resolution, yet short enough to capture as stationary significant features of the utterance.

Extensive observation of speech waveforms has revealed that the duration of stationary speech events varies over a wide range. Very short events (such as those corresponding to the burst associated with a plosive consonant) with a duration of only a few milliseconds and very long events (such as those corresponding to the production of a vowel) with a duration exceeding 350 milliseconds may be observed. Most stationary speech events have a duration in the range of 12 milliseconds to 174 milliseconds; the distribution is skewed so that the median duration (50 to 75 milliseconds) is always shorter than the mean duration (60 to 85 milliseconds). Most systems that analyze speech do so on a fixed time scale (about 20 to 25 milliseconds) that is considerably shorter than the median duration of stationary speech events and without regard for the location of the event relative to the analysis interval. Some systems employ overlapped analysis intervals (with an advance of about 10 milliseconds) so that the deleterious effects of employing a fixed time scale and ignoring event location are reduced. We have chosen to segment the utterances into 22.5 msec. blocks with an advance of 10 msec. Thus, each segment consists of 180 samples (at an 8 kHz sample rate) with each segment advanced by 80 samples. A typical utterance may be blocked into as many as 60 segments. Each segment is windowed by a Hamming window function for purposes of spectral smoothing.

IA 72 367

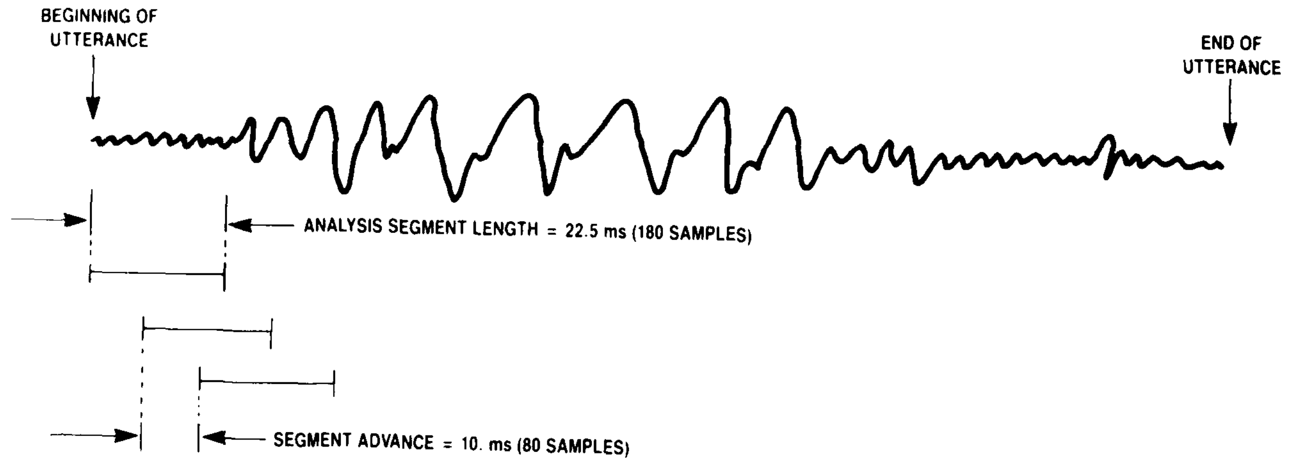

BEGINNING OF UTTERANCE
END OF UTTERANCE
ANALYSIS SEGMENT LENGTH = 22.5 ms (180 SAMPLES)
SEGMENT ADVANCE = 10. ms (80 SAMPLES)

Figure 1-4. SEGMENT EXTRACTION AND WINDOWING

1.4 SPECTRAL PROCESSING FOR FEATURE EXTRACTION

Generally, speech analysis is based on identification of spectral features that form a time-varying pattern, or spectrogram, to distinguish utterances. Short-time spectral analysis is used as a means of dealing with speech segments over time intervals in which the spectra are stationary, the concatenation of these spectral segments forming the spectrogram. The spectral approach to speech analysis is justified by the results of many years of experimentation and empirical observations and in the ability to accurately model the vocal tract and its excitation with acoustical transmission-line models. Experimental evidence abounds to show that much of the information contained, or perceived, in a speech signal is coded by the collusion of a few formants, or natural resonant frequencies, of the vocal tract.

Three methods of short-time spectral analysis are prevalent in current speech recognition research: windowed discrete Fourier analysis, processing in a band of contiguous frequency-selective filters and spectral analysis based on linear predictive coding (LPC).

Windowed discrete Fourier analysis is accomplished by computing a set of time-overlapping discrete Fourier transforms of finite length, the number of points being determined largely by the computational resources available. The filter-bank approach to spectral analysis requires the signal being analyzed to be processed by a set of bandpass filters. The highest spectral resolution is normally provided at the low-frequency end of the spectrum and lower resolution, or larger bandwidth, is provided at the upper frequencies.

Linear predictive coding (LPC) is based on a model of the utterances produced by the vocal tract as a linear system driven by white noise or a pitch-synchronous, almost periodic pulse train. Based on Wiener's (1949) work on linear prediction of stochastic time-series, a number of recursive algorithms are available to determine the parameters of the time-varying linear system, from which it is a simple matter to evaluate the system function in the frequency domain to generate the spectrogram. Alternatively, the LPC parameters can be used directly as a means of encoding speech for pattern discrimination and recognition.

The LPC method is equivalent to autoregressive spectral estimation, which in turn is equivalent to maximum entropy spectral estimation for Gaussian processes. We have chosen to use this method in our RNS development for several reasons. One is the successful use over many years of linear prediction theory for speech, which is largely due to the natural fit of an all-pole model to the speech signal during voicing, attributed to the absence of zeros in the transfer function of the transmission-line model. Another consideration is the maximum entropy viewpoint which does not constrain artificially the data where it is not observed but rather produces a spectrum that presumes maximum uncertainty for the unobserved data. Finally, the computations involved in processing the data involve autocorrelation estimates rather than spectral filtering or transforms, and these are well suited to RNS computation, particularly with RNS systolic architectures that have been developed for transversal filtering under MITRE's Integrated Electronics project.

1.4.1 Autoregressive Spectral Estimation

Autoregressive spectral estimation of speech is treated thoroughly in the literature, but its essentials will be reviewed here for completeness [1]. The autoregressive (AR) or linear predictive coding (LPC) model of speech assumes that speech may be modeled as the output of a linear system of finite order having only poles in its frequency domain transfer function and driven either by Gaussian white noise, or by a pitch-synchronous periodic signal, depending on whether the sound is unvoiced (as for certain consonants) or voiced (as for vowels). The parameters of the model are estimated, from which it is a simple matter to generate the power density spectrum.

The most common description of the AR model is in terms of the model gain, $\sigma > 0$, and a set of predictor coefficients $\{a_n; n = 1, 2, \ldots, P\}$ which are selected so that a monic P^{th} order polynomial, $z^P A_P(z)$, defined by

$$A_P(z) = \sum_{n=0}^{P} a_n z^{-n}; \quad a_0 = 1 \tag{1.1}$$

has all its roots inside the unit circle $z = e^{i\theta}$. With these parameters so defined, the P^{th} order autoregressive, or AR(P), model spectrum is given by

$$\psi(\theta) = \frac{\sigma^2}{A_P(e^{i\theta})A_P(e^{-i\theta})} \tag{1.2}$$

The requirement that $z^P A_P(z)$ have all its roots inside the unit circle is made so that the predictor coefficient sequence is unique. Note that for any root of $z^P A_P(z)$ which is inside the unit circle, there is a corresponding root of $z^{-P} A_P(z^{-1})$ which is outside the unit circle. By swapping corresponding roots between this pair of polynomials, one obtains different sets of polynomial coefficients without affecting $\psi(\theta)$. Among these 2^P polynomials, the one with all its roots inside the unit circle is called a minimum phase polynomial.

In addition to providing a unique parametric description of the AR(P) model spectrum, $\psi(\theta)$, the minimum phase condition is important in that it is equivalent to stability for the linear shift invariant filter with transfer function

$$\Psi(z) = \frac{\sigma}{A_P(z)} \tag{1.3}$$

$A_P(z)$ is often described recursively in terms of a sequence of reflection coefficients. Thus

$$A_n(z) = A_{n-1}(z) + K_n z^{-n} A_{n-1}(z^{-1}); \quad A_0(z) = 1 \tag{1.4}$$

for n = 1, 2, ..., P. If $|K_n| < 1$ and $A_{n-1}(z)$ is minimum phase, then $A_n(z)$ is also minimum phase. Thus, $A_P(z)$ is minimum phase if and only if every reflection coefficient in the set $\{K_n;\ n = 1, 2, \ldots, P\}$ is less than one in absolute value. One virtue of reflection coefficients is that they admit this simple test for the minimum phase condition.

By expanding $\psi(\theta)$ in a Fourier series, one obtains the following pair of relationships between the AR(P) model and its autocorrelation coefficient sequence.

$$\psi(\theta) = \sum_{-\infty}^{+\infty} r_n e^{-in\theta} \qquad (1.5a)$$

$$r_n = \int_{-\pi}^{\pi} \psi(\theta) e^{in\theta} \frac{d\theta}{2\pi} \qquad (1.5b)$$

Because $\psi(\theta)$ is symmetric, $r_n = r_{-n}$. By equating the right-hand side of equations 1.2 and 1.5a and then multiplying both sides of the resulting equation by $Ap(e^{i\theta})$ one obtains

$$Ap(e^{i\theta}) \sum_{-\infty}^{+\infty} r_n e^{-in\theta} = \frac{\sigma^2}{Ap(e^{-i\theta})} \qquad (1.6)$$

Then, by expanding both sides of this equation and equating coefficients of like powers of $e^{i\theta}$, one may derive the Yule-Walker, or normal, equations expressed in matrix form as

$$\begin{bmatrix} r_0 & r_1 & r_2 & \cdots & r_P \\ r_1 & r_0 & r_1 & \cdots & r_{P-1} \\ r_2 & r_1 & r_0 & \cdots & r_{P-2} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ r_P & r_{P-1} & r_{P-2} & \cdots & r_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ \cdot \\ \cdot \\ \cdot \\ a_P \end{bmatrix} = \begin{bmatrix} \sigma^2 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} \tag{1.7}$$

Given the truncated sequence of autocorrelation coefficients $\{r_n;$ $n = 0, 1, \ldots, P\}$ the above symmetric Toeplitz coefficient matrix is known, and one may solve equation (1.7) to determine the predictor coefficients. For this parameter set, the minimum phase condition is equivalent to the condition that all principal minors of the coefficient matrix have a positive determinant.

To organize the distortion function computation, to be dicussed below, it is useful to define a sequence of inverse correlation coefficients $\{u_n;\ n = 0, 1, \ldots, P\}$ as

$$\frac{1}{\psi(\theta)} = \sum_{-P}^{P} u_{|n|} e^{-in\theta} \tag{1.8}$$

These are related to the parameters of the AR model by the equation,

$$u_n = \sum_{\ell=0}^{P-n} a_\ell a_{\ell+n}/\sigma^2 \tag{1.9}$$

1.4.2 The Itakura-Saito Distortion Function

The Itakura-Saito distortion function, upon which our work is based, was introduced in 1970 as an analysis technique for making a maximum likelihood estimate of the power spectral density of an autoregressive process modeled as the output of an all-pole filter driven by white Gaussian noise [2]. Their original work has been subsequently extended by a number of researchers, and variations of their original idea have resulted in a number of related distortion measures [3]. Below, we present the basic formulas that result when the Itakura-Saito distortion is interpreted as a measure of spectral matching. In section 2, we will present a different and self-contained derivation that is developed in the time domain.

Defining $f(\theta)$ as the power spectral density of a test segment and $g(\theta)$ as the power spectral density of a reference utterance, the Itakura-Saito distortion may be expressed as

$$d_{IS}(f,g) = \int_{-\pi}^{\pi} \left(\frac{f(\theta)}{g(\theta)} - \ln\frac{f(\theta)}{g(\theta)} - 1\right)\frac{d\theta}{2\pi} \qquad (1.10)$$

When the spectra of the test and reference segments are the same, the distortion is zero for an all-pole model spectrum and it is possible to show by means of contour integration of equation (1.2) in the complex plane that

$$\int_{-\pi}^{\pi} \ell n\psi(\theta)\,\frac{d\theta}{2\pi} = \ell n\sigma^2 \tag{1.11}$$

where σ^2 is the variance of the white Gaussian process driving the all-pole filter whose transfer function is $A_P(z)$. The distortion may then be written as

$$d_{IS} = \int_{-\pi}^{\pi} \frac{f(\theta)}{g(\theta)}\,\frac{d\theta}{2\pi} - \ell n\sigma_f^2 + \ell n\sigma_g^2 - 1 \tag{1.12}$$

In terms of the inverse correlation coefficients of equation (1.8), the remaining integral may be rewritten as

$$\int_{-\pi}^{\pi} \frac{f(\theta)}{g(\theta)}\,\frac{d\theta}{2\pi} = \sum_{n=-P}^{P} u_n(g) \int_{-\pi}^{\pi} f(\theta)e^{in\theta}\,\frac{d\theta}{2\pi} \tag{1.13}$$

The integral on the right-hand side of equation (1.13) is simply the sequence of autocorrelation coefficients $\{r_n;\ n = 0, \pm1, \ldots, \pm\infty\}$ whose discrete Fourier transform is the power density spectrum $f(\theta)$, as in equation (1.5). Then we have

$$\int_{-\pi}^{\pi} \frac{f(\theta)}{g(\theta)}\,\frac{d\theta}{2\pi} = \sum_{n=-P}^{P} u_n(g)r_n(f) \tag{1.14}$$

which is seen to be a scalar product between the vector of 2P + 1 autocorrelation coefficients $\underline{r}$ with the vector of inverse correlation coefficients $\underline{u}$. It is a fortunate consequence of the autoregressive model that only 2P + 1 autocorrelation coefficients of the test process are needed in the distortion computation (and they are symmetric). When the process actually is P^{th} order autoregressive, these are sufficient to predict the remaining coefficients and hence the power density spectrum. The distortion function may finally be expressed as

$$d_{IS}(f,g) = \sum_{n=-P}^{P} u_n(g) r_n(f) - \ln \frac{\sigma_f^2}{\sigma_g^2} - 1 \tag{1.15}$$

This is the form of the Itakura-Saito distortion used for computation. In section 2 we will discuss determination of the variances σ_f^2 and σ_g^2.

In computing the distortion of equation (1.15), notice that $\underline{u}$, the vector of inverse correlation coefficients and the corresponding variance σ_g^2 for each reference segment may be precomputed from the normal equations (1.7) and equation (1.9) and stored in the reference library. For the segments of a test utterance, it is necessary to determine $\underline{r}$, the vector of 2P + 1 autocorrelation coefficients and the variance σ_f^2 since these cannot be computed in advance.

1.5 DYNAMIC TIME-WARPING

Although voice spectrograms provide distinct patterns for discrimination and recognition, variations in speaking rate, speaker inflections, and variations from speaker to speaker produce local time variations in the patterns that must be compensated for effective comparison with the stored patterns. Dynamic programming, introduced for speech recognition by Sakoe and Chiba [4], has been successfully employed to bring about adequate time registration, but the computational complexity is high because of the need to register the segments of each test utterance against those of every reference utterance, including variations of the same word, stored in the reference library.

The dynamic time-warping algorithm and its RNS implementation will be discussed in entirety in section 3. A systolic architecture for the RNS implementation will be discussed in section 4.

1.6 SUMMARY OF RESULTS

The diagram of a speech recognition system based on linear predictive coding (or autoregressive spectral estimation) and dynamic time-warping is shown in figure 1.5. This diagram corresponds to a computer simulation model used extensively in our RNS implementation studies. (The reader is referred to Appendix A for a discussion of residue number systems and their properties.)

1A-72 30C

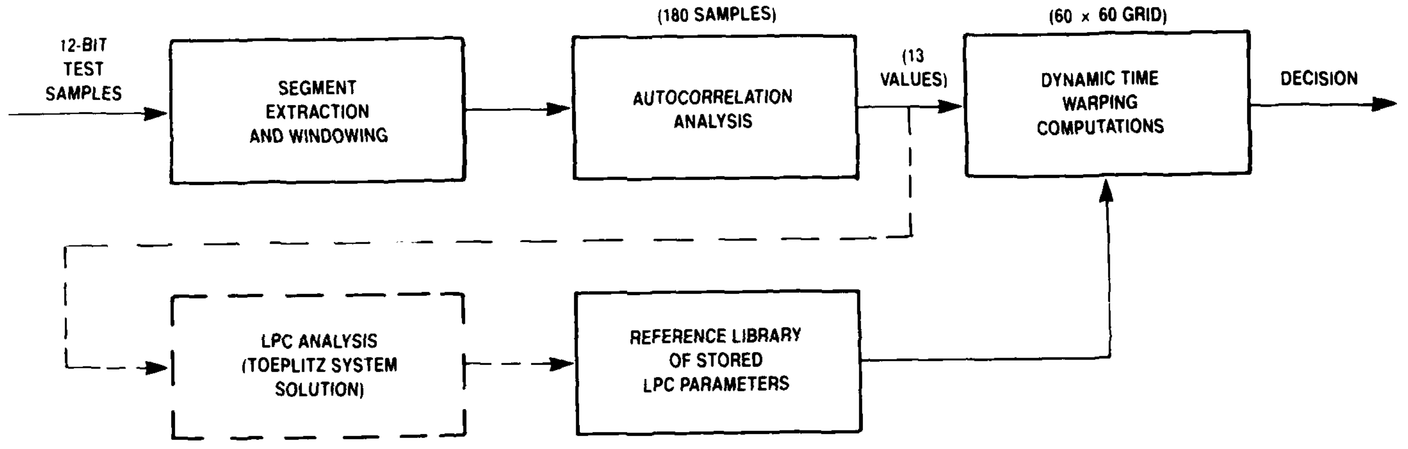

12-BIT TEST SAMPLES
SEGMENT EXTRACTION AND WINDOWING
(180 SAMPLES)
AUTOCORRELATION ANALYSIS
(13 VALUES)
(60 × 60 GRID)
DYNAMIC TIME WARPING COMPUTATIONS
DECISION
LPC ANALYSIS (TOEPLITZ SYSTEM SOLUTION)
REFERENCE LIBRARY OF STORED LPC PARAMETERS

Figure 1-5. LPC/DTW WORD RECOGNITION SYSTEM

As shown in figure 1.5, the speech waveform after preprocessing is converted to 12-bit digital samples and segmented into overlapping blocks of 180 samples each which are windowed with a Hamming window function. Thirteen autocorrelation values are estimated for each segment. The LPC parameters are extracted by solution of the normal equations (1.7) with the process gain σ_g^2 and the inverse correlation coefficients u_n stored in the reference library. The LPC computations are performed off-line and computed with floating point arithmetic, but the scaled and quantized parameters may be stored in RNS code.

For the test samples, the same autocorrelation is performed with the RNS values entered into the dynamic time warping processor to be compared with the reference library segments. The Itakura-Saito distortions are computed largely in RNS, as discussed in section 3, to establish the DTW grid. The array of 60×60 points is a typical value; actual utterances in our library range between 28 and 72 segments. The dynamic time warping path metric calculations are also carried out in RNS, as will be discussed in section 3.

Extensive studies were made with this simulation model to support our RNS implementation development and selection of RNS parameters. Our studies have shown that RNS can be quite useful in implementing a dynamic time-warping based speech recognition algorithm, although some significant problems still remain.

Our architectural studies demonstrate that RNS computation is quite natural for computing the correlation estimates needed in the LPC analysis and in the Itakura-Saito distortion computation. In fact, a linear systolic architecture can make use of hardware techniques already developed for RNS implementation of a transversal equalizer under MITRE's Integrated Electronics project. The pipelined architecture will be discussed in section 4.

We have found that the dynamic time-warping calculations can be profitably carried out in RNS if the Itakura-Saito distortion values are first quantized into the range of a smaller RNS. In fact, binary quantization (match or no-match) seems adequate if the unquantized distortion values provide sufficient discrimination between true and false word matches. We developed an algorithm for performing the quantization within RNS, which is presented in section 3, with the hardware implementation discussed in section 4.

Subject to these conditions, RNS seems useful both in computing the distortion values and in accumulating the least-cost path metric in the dynamic time-warping algorithm. In section 4, it is shown how both these computations may be carried out in a pipelined two-dimensional systolic array, with the data flow specified and implications for RNS implementation of the hardware discussed.

Although the path computations for DTW can be easily carried out in a practical-sized RNS with appropriate quantization of the distortion function values, the RNS range used to compute the Itakura-Saito distortion is still rather large (30 bits equivalent). The quantization algorithm also seems more complicated than necessary. Improvements to alleviate these conditions, and therefore to develop a more practical method of RNS implementation, are the subject of continuing work, as will be discussed in our concluding section 5.

SECTION 2

USE OF THE ITAKURA-SAITO DISTORTION FUNCTION FOR SPEECH RECOGNITION

2.1 INTRODUCTION

The Itakura-Saito distortion was derived originally for maximum likelihood estimation of the parameters of an all-pole filter used for speech synthesis [2]. In their work, it was used in an experimental system for speech analysis and resynthesis, and it demonstrated good performance by comparison of sound spectrograms of the input and resynthesized waveforms. They interpreted their distortion function as having physical meaning for comparing power density spectra of short-time speech records, a view which has appealed to the speech processing community and which has been expanded in the more recent literature [3].

Unquestionably, spectral models are useful and have a long history of use in the analysis of speech. They provide convenient mental pictures that can be related to acoustic models of the vocal tract and its excitation. Traditionally, speech patterns are represented by sound spectrograms presented as two-dimensional intensity plots of time-varying power spectra. Trained speech researchers learn to "read" such plots, demonstrating the ability to recognize patterns not readily apparent in the temporal waveform. It is not surprising, therefore, that speech recognition work is so heavily influenced by the spectral viewpoint.

On the other hand, the Itakura-Saito distortion function is well-suited to formulation in the time domain, such a formulation being relatively simple and free from such sophisticated mathematical notions as the asymptotic distribution of eigenvalues associated with Toeplitz forms, results of which are needed in a rigorous spectral approach [5]. Itakura and Saito modeled speech as an autoregressive random process produced at the output of an all-pole linear filter driven by white Gaussian noise [6]. They showed that this model, with its parameters obtained as a maximum-likelihood estimate, minimized their distortion function. In an efficient distortion computation based on a linear predictive coding model, the frequency spectra are not explicitly used, and in fact, an appropriate distortion function can be derived entirely in the time domain. In an application such as word recognition, a frequency-independent approach may have merit in illuminating the essential computational aspects. The temporal approach also leads naturally to a matched-filtering interpretation which suggests an alternative computational implementation.

2.2 A FREQUENCY-INDEPENDENT FORMULATION

We begin by expressing the distortion function as an average log-likelihood ratio, which measures the distinguishability between two random processes f and g [7].

$$d_{IS} = \int_{R^n} p(\underline{x}\,|f)\,\ell n \frac{p(\underline{x}|f)}{p(\underline{x}|g)}\, d\underline{x}. \tag{2.1}$$

The symbols f and g represent zero-mean Gaussian random processes corresponding to test and reference utterances, respectively. In application to speech recognition, since the samples $\underline{x}$ are drawn from the process f, the conditional probability density functions are related by the inequality $p(\underline{x}|g) \leq p(\underline{x}|f)$, therefore $d_{IS} \geq 0$.

Samples of the Gaussian process f can always be regarded as linear combinations of samples of a white Gaussian noise process, in other words, as the output of a stable discrete linear system driven by samples of white Gaussian noise. The linear system may be described in general by the linear difference equation,

$$x(n) + \sum_{\ell=1}^{L} \alpha_\ell x(n-\ell) = \sigma_f \sum_{k=0}^{K} \beta_k \mu(n-k) \tag{2.2}$$

where $L \leq K$, σ_f^2 is the variance of the input white noise process, $\beta_0 = 1$ and x(n) represents the sequence of output samples. For the process g, we assume that the corresponding linear system is described by the restricted difference equation

$$s(n) + \sum_{m=1}^{M} a_m s(n-m) = \sigma_g \mu(n). \tag{2.3}$$

In (2.3), σ_g^2 is the variance of the input process and s(n) represents the sequence of output samples. Notice that, unlike x(n), s(n) depends only on the present and not on past input samples, and is referred to as an autoregressive random process (the parameters a_m determine an all-pole filter in accordance with the frequency-domain transfer function corresponding to (2.3)). While σ_g^2 is the variance of the input process, it will be convenient to regard σ_g as the gain of the linear system driven by $\mu(n)$, (n = 0, 1, 2, ...), a sequence of samples of unit-variance white Gaussian noise.

We assume that the distortion is to be computed for a sequence of N samples x(n) of the test process f, and that the comparison is to be made with the model of the reference process g that is described by the linear system of (2.3). We denote by $\underline{x}$ the column vector of N samples $[x(0), x(1), \ldots, x(N-1)]^t$ and by $\underline{R}_f$ the $N \times N$ matrix $[r_{mn}]$ of (ensemble-average) covariance values of $\underline{x}$. Similarly, we denote by $\underline{S}_g$ the covariance matrix of N samples of the process g and we assume that $M < N$. With this notation, we may express the conditional Gaussian probability density functions as [8],

$$p(\underline{x}|f) = \frac{1}{(2\pi)^{N/2}|\underline{R}_f|^{1/2}} \exp\left(-\frac{1}{2}\underline{x}^t\underline{R}_f^{-1}\underline{x}\right) \qquad (2.4)$$

and

$$p(\underline{x}|g) = \frac{1}{(2\pi)^{N/2}|\underline{S}_g|^{1/2}} \exp\left(-\frac{1}{2}\underline{x}^t\underline{S}_g^{-1}\underline{x}\right) \qquad (2.5)$$

where $|\underline{R}|$ signifies determinant($\underline{R}$). After substitution of (2.4) and (2.5) into (1), the distortion may be expressed as,

$$d_{IS} = \frac{1}{2}\ln\frac{|\underline{S}_g|}{|\underline{R}_f|} - \frac{1}{2}\,\overline{\underline{x}^t\underline{R}_f^{-1}\underline{x}}^{\,f} + \frac{1}{2}\,\overline{\underline{x}^t\underline{S}_g^{-1}\underline{x}}^{\,f} \tag{2.6}$$

where the overbar signifies the ensemble average over the multivariate distribution of $\underline{x}$ conditioned on f.

The second term of (2.6) may be expanded via

$$\overline{\underline{x}^t\underline{R}_f^{-1}\underline{x}}^{\,f} = \frac{1}{|\underline{R}_f|}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}|\underline{R}_f|_{mn}\,\overline{x_m x_n}^{\,f} \tag{2.7}$$

where $|\underline{R}_f|_{mn}$ is the cofactor of the element r_{mn} in the determinant of the covariance matrix $\underline{R}_f$. We recognize the inner sum as Laplace's expansion of $|\underline{R}_f|$, consequently,

$$\overline{\underline{x}^t\underline{R}_f^{-1}\underline{x}}^{\,f} = N \tag{2.8}$$

and

$$d_{IS} = \frac{1}{2}\ln\frac{|\underline{S}_g|}{|\underline{R}_f|} - \frac{N}{2} + \frac{1}{2}\,\overline{\underline{x}^t\underline{S}_g^{-1}\underline{x}}^{\,f}. \tag{2.9}$$

To compute the ensemble average remaining in (2.9), we invoke the properties of the linear system of (2.3). There exists a linear system that is inverse to that of (2.3) in the sense that it is a "whitening" filter for s(n). In other words, if the two systems are cascaded and driven by white Gaussian noise, then the output is also white Gaussian noise. The unit impulse response of the cascaded system must be a unit impulse. If one writes out a few terms of the recursion of (2.3) for $x(0) = 0$, and $\mu(0) = 1$, $\mu(n > 0) = 0$, then it is almost trivial to verify that the inverse filter is a finite impulse response filter with its impulse response given by the sequence

$$[h(0), h(1), \ldots, h(M)] = [1, a_1, a_2, \ldots, a_M]/\sigma_g. \qquad (2.10)$$

If we let $\underline{\nu}$ represent an N-vector $[\nu(0), \nu(1), \ldots, \nu(N-1)]^t$ of output samples of the inverse filter, subject to (unit-variance) white Gaussian noise input $\underline{\mu} = [\mu(0), \mu(1), \ldots, \mu(N-1)]$, then the output covariance matrix, $\overline{\underline{W_g}} = \underline{\nu\nu^t}$, may be written as

$$\underline{W_g} = \underline{H\overline{\mu\mu^t}H^t} = \underline{HIH^t} = \underline{HH^t} \qquad (2.11)$$

where $\underline{H}$ is an $N \times N$ Toeplitz matrix composed of impulse response elements $h(k)$,

$$\underline{H} = \begin{bmatrix} h_0 & & & & & & & & & \\ h_1 & h_0 & & & & & & & & \\ h_2 & h_1 & h_0 & & & & \underline{0} & & & \\ \cdot & \cdot & \cdot & \cdot & & & & & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & & & & & \\ h_M & h_{M-1} & \cdot & \cdot & h_1 & h_0 & & & & \\ & h_M & \cdot & \cdot & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & \\ & \underline{0} & & h_M & \cdot & \cdot & \cdot & h_1 & \cdot & \\ & & & & h_M & \cdot & \cdot & \cdot & h_1 & h_0 \end{bmatrix} \qquad (2.12)$$

Since the inverse filter is a whitening filter for $\underline{s} = [s(0), s(1), \ldots, s(N-1)]^t$, we also have $\underline{\mu} = \underline{Hs}$. Consequently,

$$\overline{\underline{\mu}\,\underline{\mu}^t} = \underline{H}\,\overline{\underline{s}\,\underline{s}^t}\,\underline{H}^t = \underline{H}\,\underline{S}_g\,\underline{H}^t. \qquad (2.13)$$

Then, with reference to (2.11),

$$\underline{S}_g^{-1} = \underline{W}_g \qquad (2.14)$$

which shows the covariance matrix of the reference speech model samples to be the inverse of the covariance matrix of the output of the inverse filter (when driven by white Gaussian noise). Now, we may write

$$\overline{\underline{x}^t \underline{S}_g^{-1} \underline{x}}^f = \overline{\underline{x}^t \underline{W}_g \underline{x}}^f \qquad (2.15)$$

where $\underline{W}_g = \underline{HH}^t$ is to be determined by (2.12) from the model parameters of the reference process g.

$\underline{W}_g$ is an N × N symmetric matrix with its elements determined as

$$W_{mn} = \sum_{k=0}^{m \leq M} h_k h_{|m-n|+k} \qquad (2.16)$$

for $|m - n| \leq M$. For $|m - n| > M$, $W_{mn} = 0$. In terms of the linear system parameters, for $|m - n| \leq M$,

$$W_{mn} = \frac{1}{\sigma_g^2} \sum_{k=0}^{m \leq M} a_k a_{k+|m-n|} = \frac{1}{\sigma_g^2} A_{mn} \qquad (2.17)$$

where, by definition, $a_0 = 1$. It is convenient to regard $\underline{W}_g$ as the sum of two matrices,

$$
\underline{W}_g = \left[\begin{array}{cccccccccc}
 & & & & & W_M & & & & \\
 & & & & & W_{M-1} & W_M & & & \\
 & \underline{0} & & & & \cdot & \cdot & & \underline{0} & \\
 & & & & & \cdot & \cdot & & & \\
 & & & & & \cdot & \cdot & & & \\
W_M & \cdot & \cdot & \cdot & W_1 & W_0 & W_1 & \cdot & W_M & \\
 & W_M & \cdot & \cdot & \cdot & W_1 & W_0 & W_1 & \cdot & W_M \\
 & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & W_1 \\
 & \underline{0} & & & W_M & \cdot & \cdot & \cdot & W_1 & W_0
\end{array}\right] +
$$

$$
\left[\begin{array}{cccccc}
W_{00} & W_{01} & \cdot & \cdot & W_{0,M-1} & \\
W_{10} & W_{11} & \cdot & \cdot & W_{1,M-1} & \\
\cdot & & & \cdot & \cdot & \underline{0} \\
\cdot & & & \cdot & \cdot & \\
W_{M-1,0} & W_{M-1,1} & \cdot & \cdot & W_{M-1,M-1} & \\
 & \underline{0} & & & & \underline{0}
\end{array}\right] \qquad (2.18)
$$

For m or n $\geq$ M, we have expressed $W_{mn} = W_{|m-n|}$ in (2.18).

Except for the upper left hand M × M block, $\underline{W}_g$ would be a symmetric Toeplitz matrix, the form of a covariance matrix for a wide-sense stationary process. It may seem odd that $\underline{W}_g$ is not Toeplitz, but not if it is observed that $\underline{W}_g$ is the covariance matrix of the output samples of a linear system driven by a stationary input, with zero-state initial conditions. The output of this system is eventually stationary, after the initial transient settles, but it is strictly non-stationary even though the input is stationary [9].

If we denote by $\underline{r}$ the vector of 2M + 1 elements $\underline{r} = [r_M, r_{M-1}, \ldots, r_1, r_0, r_1, \ldots, r_M]$ and similarly define $\underline{A} = [A_M, A_{M-1}, A_1, A_0, A_1, \ldots, A_M]$, then the bilinear form (2.15) can be written in terms of a scalar product

$$\overline{\underline{x}^t \underline{W}_g \underline{x}}^{f} = \frac{N-M}{\sigma_g^2} (\underline{A} \cdot \underline{r}) + \frac{1}{\sigma_g^2} \sum_{k=0}^{M-1} \sum_{\ell=1}^{M-2k} (m - k - \ell) r_k a_{\ell-1} a_{\ell+k-1}. \qquad (2.19)$$

Now we have,

$$d_{IS} = \frac{1}{2} \ell n \frac{|\underline{S}_g|}{|\underline{R}_f|} - \frac{N}{2} + \frac{N-M}{2\sigma_g^2} (\underline{A} \cdot \underline{r}) + \qquad (2.20)$$

$$\frac{1}{2\sigma_g^2} \sum_{k=0}^{M-1} \sum_{\ell=1}^{M-2k} (m-k-\ell) r_k a_{\ell-1} a_{\ell+k-1}.$$

For M << N, a good approximation is simply,

$$d_{IS} \simeq \frac{1}{2} \ell n \frac{|\underline{S}_g|}{|\underline{R}_f|} - \frac{N}{2} + \frac{N}{2\sigma_g^2} (\underline{A} \cdot \underline{r}). \qquad (2.21)$$

We will be concerned next with computation of the determinants $|\underline{S}_g|$ and $|\underline{R}_f|$. Direct computation of the N × N determinants, by using Laplace's expansion for example, would impose an enormous computational burden since N is typically greater than a hundred samples. But, since $|\underline{S}_g| = |\underline{W}_g|^{-1}$, use of the parameters of the all-pole filter model can save considerable work. Since $\underline{W}_g = \underline{HH}^t$, the product of lower- and upper-triangular matrices, and $\det(\underline{HH}^t) = \det(\underline{H})\det(\underline{H}^t) = h_0^{2N}$, it follows immediately from (2.10) that

$$\ell n|\underline{S}_g| = N\ \ell n\ \sigma_g^2. \tag{2.22}$$

As noted earlier, σ_g^2 is the variance of the white noise process driving the linear system of (2.3). It can be determined readily from the normal equations which must be solved to determine the system parameters [1].

$$\begin{bmatrix} S_0 & S_1 & S_2 & \cdots & S_M \\ S_1 & S_0 & S_1 & \cdots & S_{M-1} \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ S_M & S_{M-1} & S_{M-2} & \cdots & S_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \\ \cdot \\ \cdot \\ \cdot \\ a_M \end{bmatrix} = \begin{bmatrix} \sigma_g^2 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} \tag{2.23}$$

as the scalar product of the first row of $\underline{S}_g$ with the vector of filter coefficients,

$$\sigma_g^2 = \underline{S}_M^t \underline{a}_M. \tag{2.24}$$

The vector $[S_0, S_1, \ldots, S_M] = \underline{S}_M^t$ consists of the first M values of the covariance of N samples of the speech reference process, assumed sampled in a stationary interval.

If we are willing to model the test process also as an autoregressive process of order M, then we can similarly determine σ_f^2 from the normal equations, but it is not necessary to actually solve them. Let

$$\begin{bmatrix} r_0 & r_1 & \cdot & \cdot & \cdot & r_M \\ r_1 & r_0 & \cdot & \cdot & \cdot & r_{M-1} \\ \cdot & & & & & \cdot \\ \cdot & & & & & \cdot \\ \cdot & & & & & \cdot \\ r_M & r_{M-1} & \cdot & \cdot & \cdot & r_0 \end{bmatrix} \begin{bmatrix} 1 \\ b_1 \\ \cdot \\ \cdot \\ \cdot \\ b_M \end{bmatrix} = \begin{bmatrix} \sigma_f^2 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} \tag{2.25}$$

be the normal equations defining the parameters of the test process (assumed autoregressive). Application of Cramer's rule to (2.25) yields,

$$\sigma_f^2 = \frac{|\underline{R}_f^{(M)}|}{|\underline{R}_f^{(M-1)}|} \tag{2.26}$$

where $|\underline{R}_f^{(M)}|$ is the determinant of the (M+1) × (M+1) covariance matrix and $|\underline{R}_f^{(M-1)}|$ is the cofactor of the first element. In this case, rather than computing an N × N determinant, the autoregressive assumption for the test process allows the computation of the ratio of determinants of substantially smaller order, since generally $M \ll N$.

Finally, we may express the distortion as (approximately, for $M \ll N$)

$$d_{IS} \simeq \frac{1}{2} \ln \frac{\sigma_g^2}{\sigma_f^2} - \frac{N}{2} + \frac{N}{2\sigma_g^2} (\underline{A} \cdot \underline{r}). \tag{2.27}$$

If we define the normalized covariance coefficients for the test process as $\rho_k = r_k/\sigma_f^2$, then the approximate distortion function (2.27) is expressed as

$$d_{IS} \simeq \frac{1}{2} \ln \frac{\sigma_g^2}{\sigma_f^2} - \frac{N}{2} + \frac{N}{2} \frac{\sigma_f^2}{\sigma_g^2} (\underline{A} \cdot \underline{\rho}). \tag{2.28}$$

If the processes are identical, except for the input variances, then (2.28) becomes

$$d_{IS} \simeq \frac{1}{2} \ln \frac{\sigma_g^2}{\sigma_f^2} - \frac{N}{2} \left(1 - \frac{\sigma_f^2}{\sigma_g^2}\right) \tag{2.29}$$

which would vanish for equal input variances.

2.2.1 Gain Normalization

For purposes of word recognition, we would like the distortion function to be independent of relative differences between the model input variances of the test and reference utterances. We would like to recognize words produced by the same LPC model regardless of intensity differences. For development of such a distortion function, it is appropriate to use a log-likelihood ratio based on normalized Gaussian probability density functions.

Let

$$\begin{aligned} \underline{\beta} &= \underline{x}/\sigma_f \\ \tilde{\underline{R}}_f &= \underline{R}_f/\sigma_f^2 \end{aligned} \tag{2.30}$$

and consequently,

$$|\tilde{\underline{R}}_f| = |\underline{R}_f|/\sigma_f^{2N}. \tag{2.31}$$

For the probability density of $\underline{\beta}$,

$$p(\underline{\beta}|f) = \sigma_f^N p(\underline{x}|f) = \frac{1}{(2\pi)^{N/2}|\underline{\tilde{R}}_f|^{1/2}} \exp\left(-\frac{1}{2}\underline{\beta}^t\underline{\tilde{R}}_f^{-1}\underline{\beta}\right) \qquad (2.32)$$

If $\underline{y}$ is a sample vector of the process g, then it has a probability density function given by

$$p(\underline{y}|g) = \frac{1}{(2\pi)^{N/2}|\underline{S}_g|^{1/2}} \exp\left(-\frac{1}{2}\underline{y}^t\underline{S}_g^{-1}\underline{y}\right) \qquad (2.33)$$

and if we let

$$\begin{aligned} \underline{\alpha} &= \underline{y}/\sigma_g \\ \underline{\tilde{S}}_g &= \underline{S}_g/\sigma_g^2 \end{aligned} \qquad (2.34)$$

then the normalized probability density function is

$$p(\underline{\alpha}|g) = \frac{1}{(2\pi)^{N/2}|\underline{\tilde{S}}_g|^{1/2}} \exp\left(-\frac{1}{2}\underline{\alpha}^t\underline{\tilde{S}}_g^{-1}\underline{\alpha}\right). \qquad (2.35)$$

The gain-normalized distortion function can be derived from the normalized average log-likelihood ratio

$$\tilde{d}_{IS} = \int_{R^N} p(\underline{\beta}|f) \ln \frac{p(\underline{\beta}|f)}{p(\underline{\beta}|g)}\, d\underline{\beta} \tag{2.36}$$

where $\underline{\beta}$ is the gain-normalized test utterance vector described in (2.30). With reference to the previous formulation, we observe that $|\underline{\tilde{S}}_g| = |\underline{\tilde{R}}_f| = 1$ as a result of the gain normalization. Then

$$\tilde{d}_{IS} = -\frac{1}{2}\, \overline{\underline{\beta}^t \underline{\tilde{R}}_f^{-1} \underline{\beta}}^{\,f} + \frac{1}{2}\, \overline{\underline{\beta}^t \underline{\tilde{S}}_g^{-1} \underline{\beta}}^{\,f}. \tag{2.37}$$

Since $\underline{\tilde{R}}_f$ is the covariance matrix for $\underline{\beta}$, the first term is simply $-N/2$ and

$$\tilde{d}_{IS} = -\frac{N}{2} + \frac{1}{2}\, \overline{\underline{\beta}^t \underline{\tilde{S}}_g^{-1} \underline{\beta}}^{\,f} \tag{2.38}$$

or, in terms of the observed test vector $\underline{x}$

$$\tilde{d}_{IS} = -\frac{N}{2} + \frac{1}{2\sigma_f^2}\, \overline{\underline{x}^t \underline{\tilde{S}}_g^{-1} \underline{x}}^{\,f}\ . \tag{2.39}$$

From (2.27), we may write the gain-normalized distortion function as

$$\tilde{d}_{IS} \simeq -\frac{N}{2} + \frac{N}{2\sigma_f^2} (\underline{A} \cdot \underline{r}) \qquad (2.40)$$

or, in terms of the normalized test correlation coefficients ρ_k,

$$\tilde{d}_{IS} \simeq -\frac{N}{2} + \frac{N}{2} (\underline{A} \cdot \underline{\rho}). \qquad (2.41)$$

This is the most convenient form for computation of the gain-normalized Itakura-Saito distortion, expressed as the scalar product of two normalized vectors, the vector $\underline{A}$ composed of the correlation values of the parameters of the inverse filter model (the so-called inverse correlation coefficients) and those of $\underline{\rho}$ composed of values of the N normalized covariance samples of the test process. Although ensemble averages have been used in the formulations above, stationarity allows for computation with time averages, the time averages asymptotically approaching the ensemble averages for a sufficiently large number of samples, equivalent to a long time average.

2.3 A MATCHED FILTERING INTERPRETATION

Let $\underline{H}$ represent the matrix of impulse response coefficients of the inverse filter corresponding to the reference process g, as given by (2.10) and (2.12). Let $\underline{x}$, the sample vector drawn from the test process f, be the input to the transposed system $\underline{H}^t$, and let $\underline{z}$ be the vector of output samples. Then the inputs and outputs are related by the matrix equation

$$\underline{z} = \underline{H}^t\underline{x}. \tag{2.42}$$

The sum of the squares of the output samples may be expressed as an inner product

$$\underline{z}^t\underline{z} = (\underline{H}^t\underline{x})^t(\underline{H}^t\underline{x}) = \underline{x}^t\underline{H}\ \underline{H}^t\underline{x}. \tag{2.43}$$

From (2.11) and (2.14),

$$\underline{z}^t\underline{z} = \underline{x}^t\underline{S}_g^{-1}\underline{x}. \tag{2.44}$$

The term on the right hand side of (2.44) may be expressed as the trace of a matrix product [7],

$$\underline{x}^t\underline{S}_g^{-1}\underline{x} = \mathrm{tr}[\underline{S}_g^{-1}(\underline{x}\ \underline{x}^t)]. \tag{2.45}$$

Taking the ensemble average of both sides of (2.44), we obtain

$$\overline{\underline{z}^t\underline{z}}^{\,f} = \mathrm{tr}[\underline{S}_g^{-1}\overline{(\underline{x}\ \underline{x}^t)}^{\,f}] = \mathrm{tr}[\underline{S}_g^{-1}\underline{R}_f]. \qquad (2.46)$$

The vector $\underline{z}$, from (2.42), may be interpreted as a result of convolving the test sequence $\underline{x}$ with the inverse filter, or whitening filter obtained by LPC analysis, but with the samples of $\underline{x}$ entering the filter in reversed order. The average is taken over an ensemble of sample functions $\underline{x}$ drawn from the test process f. If f is a wide-sense stationary ergodic process, then the averaging may be accomplished by squaring and adding the filter output samples over a time interval that is long compared with the time intervals for which the samples of $\underline{x}$ are correlated, the effective coherence time. This condition should be satisfied for $M \ll N$.

In the expression for the gain-normalized distortion function (2.37), we may write

$$\tilde{d}_{IS} \simeq -\frac{1}{2}\mathrm{tr}[\underline{\tilde{R}}_f^{-1}\underline{\tilde{R}}_f] + \frac{1}{2}\,\mathrm{tr}[\underline{\tilde{S}}_g^{-1}\underline{\tilde{R}}_f]. \qquad (2.47)$$

Since

$$\mathrm{tr}[\underline{\tilde{R}}_f^{-1}\underline{\tilde{R}}_f] = N = \mathrm{tr}[\underline{\tilde{S}}_g^{-1}\underline{\tilde{S}}_g] \qquad (2.48)$$

and the trace is a distributive function, the distortion may also be expressed as,

$$\tilde{d}_{IS} \approx \frac{1}{2}\mathrm{tr}[\underline{\tilde{S}}_g^{-1}(\underline{\tilde{R}}_f - \underline{\tilde{S}}_g)]. \tag{2.49}$$

The distortion as expressed in (2.49) may be interpreted as the result of passing the difference sequence ($\underline{\beta} - \underline{\alpha}$), the difference between the normalized test and reference sample vectors, through the normalized inverse system $\underline{\tilde{H}}^t$ to obtain the output vector

$$\underline{\xi} = \underline{\tilde{H}}^t(\underline{\beta} - \underline{\alpha}) \tag{2.50}$$

and then forming the ensemble-averaged inner product

$$\overline{\underline{\xi}^t\underline{\xi}} = \overline{(\underline{\beta}^t - \underline{\alpha}^t)\underline{\tilde{H}\tilde{H}}^t(\underline{\beta} - \underline{\alpha})}. \tag{2.51}$$

For stationary ergodic processes, this may be accomplished by convolving the sequence ($\underline{\beta} - \underline{\alpha}$), in reversed order, with the normalized LPC model inverse filter having the impulse response values $\{1, a_1, a_2, \ldots, a_M\}$ and then squaring and adding the output samples.

A principal difference with the matched filter implementation is that the final distortion value is reached monotonically from below rather than being computed as a convergent series with sign alternation as in a direct computation of (2.41). This could be important in controlling the computational range of the processor in an integer computation. In practice it may be satisfactory to replace the squaring detector at the output with one that simply adds the magnitudes of the samples, thus compressing the integer range of the output.

2.4 COMPUTATION IN A RESIDUE NUMBER SYSTEM

The key equations for computation of the distortion are equation (2.27) for the Itakura Saito, and equation (2.41) for the gain normalized Itakura-Saito distortion. Equation (2.51) presented an alternative for computation of the gain-normalized distortion function, providing an output that increases monotonically to the final value.

In a computation using the real numbers in a conventional weighted number system such as two's-complement, the use of equation (2.51) rather than equation (2.41) could be important in containing the dynamic range of the processor. In a computation using a residue number system it makes little difference since, in RNS, intermediate products may overflow the range available provided that the eventual output is contained within the range of the RNS. (The reader is again referred to Appendix A for a discussion of residue number systems and their properties.)

If the Itakura-Saito distortion is used, then it will be necessary to compute the logarithms of the squared gains σ_f^2 and σ_g^2. It can be assumed that the reference gain term $\ln\sigma_g^2$ has been precomputed, converted to RNS representation and stored in the reference library. Similarly, solution of the normal equations to obtain the vector $\underline{A}$ of inverse correlation coefficients (or the vector $\underline{u} = \underline{A}/\sigma_g^2$ of normalized inverse correlation coefficients) can be assumed to have been precomputed, scaled and converted to RNS form before being stored in the reference library. Computation of $-\ell n\sigma_f^2$, which is needed in equation (2.27), can be obtained from equation (2.26) as

$$-\ell n\sigma_f^2 = \ell n|\underline{R}_f^{(M-1)}|-\ell n|(\underline{R}_f^{(M)}|. \qquad (2.52)$$

The determinants in equation (2.52) can be computed in RNS if the correlation values are scaled and converted to RNS, or are immediately available if they have already been computed in RNS. The logarithm, however, will necessitate conversion to a weighted number system before computation, with the result converted back to RNS. An excellent algorithm for logarithmic quantization for numbers represented in two´s complement form is contained in [10]. It results in a simple hardware implementation. Conversion to and from RNS using mixed-radix representation is described in Appendix A.

If the gain-normalized Itakura-Saito distortion is used as in equation (2.41), then the normalized correlation coefficients $\underline{\rho} = \underline{r}/\sigma_f^2$ must be computed. Although $\underline{r}$ may have been computed in RNS, reconversion to a weighted number system to facilitate the division is to be expected, after which the values can be scaled and reconverted to RNS. This additional conversion process should not be distressing since it occurs only once for each correlation vector; whereas the distortion function must be computed for each point in a constrained grid of points involved in the dynamic time warping algorithm, to be discussed in section 3.

SECTION 3

RESIDUE NUMBER SYSTEM IMPLEMENTATION OF A DYNAMIC TIME-WARPING BASED SPEECH RECOGNITION SYSTEM

3.1 INTRODUCTION

The basic operation performed by a speech recognition system (SRS) is the matching of an analyzed test pattern representing the unknown word to be identified against stored reference patterns which represent the words of the system vocabulary. A problem that arises in this matching is the need for time registration of the different speech patterns. The pattern representing one production of a word will differ in length from the pattern representing another production of the same word. Furthermore, individual parts of a word may be stretched or compressed relative to the same parts of another production of the same word. Attempts to perform linear time registration of speech patterns have been largely unsuccessful. However, time registration by dynamic programming [4,11] has proven to be an effective means for comparing unknown test patterns against stored reference patterns of speech.

A dynamic time-warping (DTW) algorithm finds a shortest path through a grid of points. Each point of the grid represents a matching of a selected pair of short-time segments, or frames, of the unknown test pattern and a given reference pattern. Associated with each grid point is a value which is the calculated local distortion for the particular match of test and reference frames represented by the point. Associated with each path through the grid is a distance which is a weighted sum of the local distortions for grid points lying on the path. The output of the DTW algorithm is a score, the distance of the shortest path through the grid, representing the degree of dissimilarity between the matched patterns.

Computationally, the most intensive portion of a DTW-based SRS is the calculation of the local distortion measures. This calculation must be performed for each point of the defined DTW grid, which typically may contain several thousand points, and must be repeated for each grid, that is, for each reference pattern contained in the library. The distortion measures employed in our work have been variants of the Itakura-Saito distortion measure [2]. This measure has been selected because it has a strong theoretical justification, is known to have performed well in existing recognition systems, and is relatively easy to compute. The basic computation involved in evaluating this measure is an inner product calculation for a pair of vectors of correlation and inverse correlation coefficients representing the test and stored reference frames, respectively. Inner product calculations are well-suited for implementation in RNS if the end result does not need to be immediately translated back from RNS to conventional arithmetic notation. This particular calculation of the Itakura-Saito distortion presents both a challenge and a considerable opportunity for RNS implementation. The challenge results from the apparent need to employ a large range in the RNS calculations to avoid certain problems of truncation error resulting from the integer conversion of the input data to the calculation. The opportunity results from the established fact that the range need not contain the (much larger) individual products or sums accumulated during the inner product calculation, nor even the scaled autocorrelation and inverse autocorrelation coefficients. Overflow of the RNS during the calculation causes no harm as long as the final result lies within the range of the RNS, and even occasional overflow by the result may not be harmful to the DTW distance computation.

Three functions are required for dynamic time-warping: construction of the DTW grid, the set of points (i(k),j(k)) on which the DTW path is permitted to lie; evaluation of a local distortion measure for all points of the grid; and solving to find the shortest path through the DTW grid from the point (1,1) to the point (m,n), where m is the number of reference frames and n is the number of test frames to be matched. The shortest path algorithm is a special simple case of dynamic programming [12]. Construction of the DTW grid and the calculations required to find the shortest path through the grid are discussed in section 3.2. Calculation of the Itakura-Saito distortion measure and its variants has been discussed previously in section 2.

The remainder of section 3 is concerned with the implementation of the DTW algorithm in RNS. The approach discussed in section 3.3 utilizes a two-part quantization of distortion values, first into the range of a single modulus, and then to a single bit (match or no-match). The quantization algorithm is described in section 3.4. RNS implementation of the shortest-path calculation is treated in section 3.5. Range considerations for the RNS implementation of dynamic time-warping are discussed in section 3.6. Finally, results of simulations of RNS implementations of a DTW-based speech recognition system are presented in the concluding section 3.7.

3.2 DYNAMIC TIME-WARPING ALGORITHM

The three functions contained in the dynamic time-warping algorithm are illustrated in figure 3.1. In this section the determination of the DTW grid point set, given the number of reference frames m, the number of test frames n, and a set of local and global path constraints, is described, and the calculations required for finding

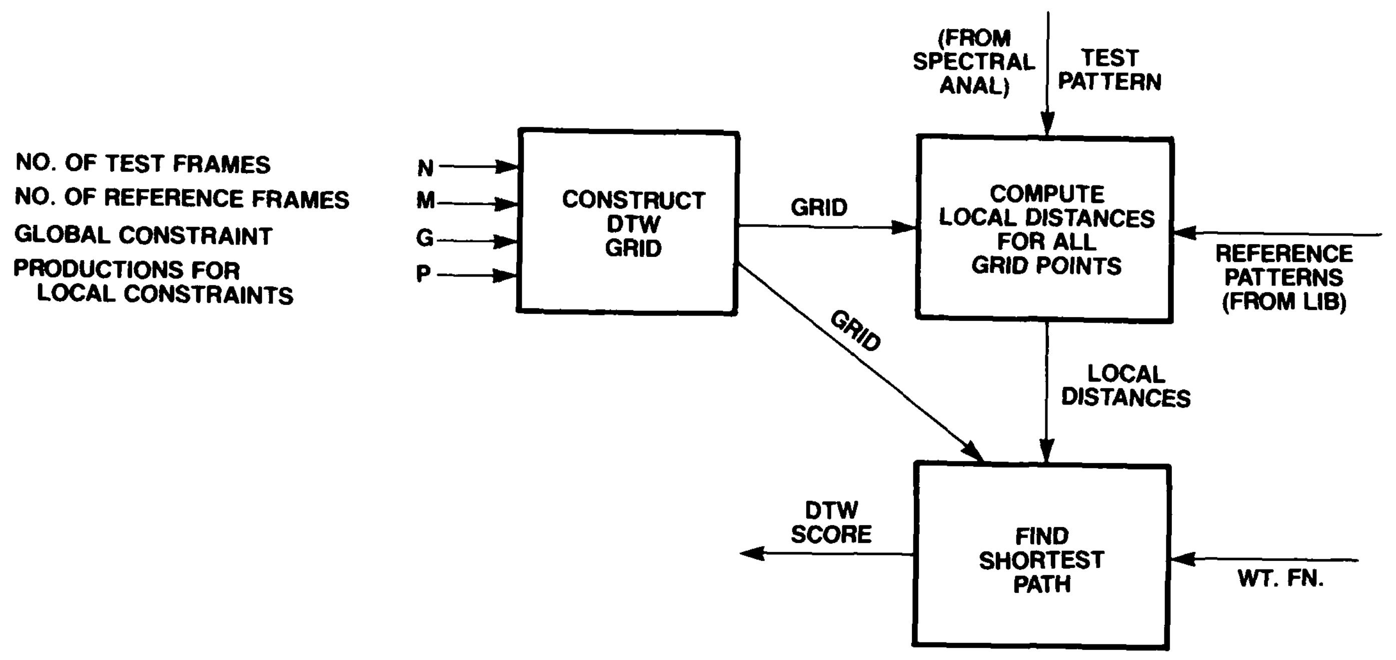


Figure 3.1 DYNAMIC TIME-WARPING

the shortest path through the grid are derived for a particular choice of local constraints. The unknown test utterance will always be assigned to the y-axis (vertical), and the reference utterance will be assigned to the x-axis (horizontal).

3.2.1 DTW Path Constraints

Initially, before application of any constraints, the DTW grid (figure 3.2) consists of the m × n points (i,j), $1 \leq i \leq m$, $1 \leq j \leq n$. Each point (i,j) represents the matching of the i-th reference frame against the j-th test frame. Certain matches and sequences of matches (i.e., paths) may be unreasonable to make, however, and should be ruled out in advance. Rules are adopted in a speech recognition system to avoid such unreasonable paths and pointless computation. It is the role of the local and global path constraints to define these rules.

Local path constraints specify in a precise manner the ways in which a particular path point (i(k),j(k)) can be reached from a preceding path point (i(k - 1), j(k - 1)). Following Myers et al. [11], we represent allowed local paths by a set of productions from a regular grammar. A production is a rule of the form

$$P: (a_1,b_1)(a_2,b_2)\dots(a_L,b_L) \qquad (3.1)$$

where L is the length of the production, and the (a,b)'s are segments in a sequence of local moves. All a's and b's and L are assumed to be (small) nonnegative integers. Using a production, a local path to the point (i(k),j(k)) can be traced backwards to the point (i(k - 1), j(k - 1)) through L - 1 intermediate points:

IA-72.529

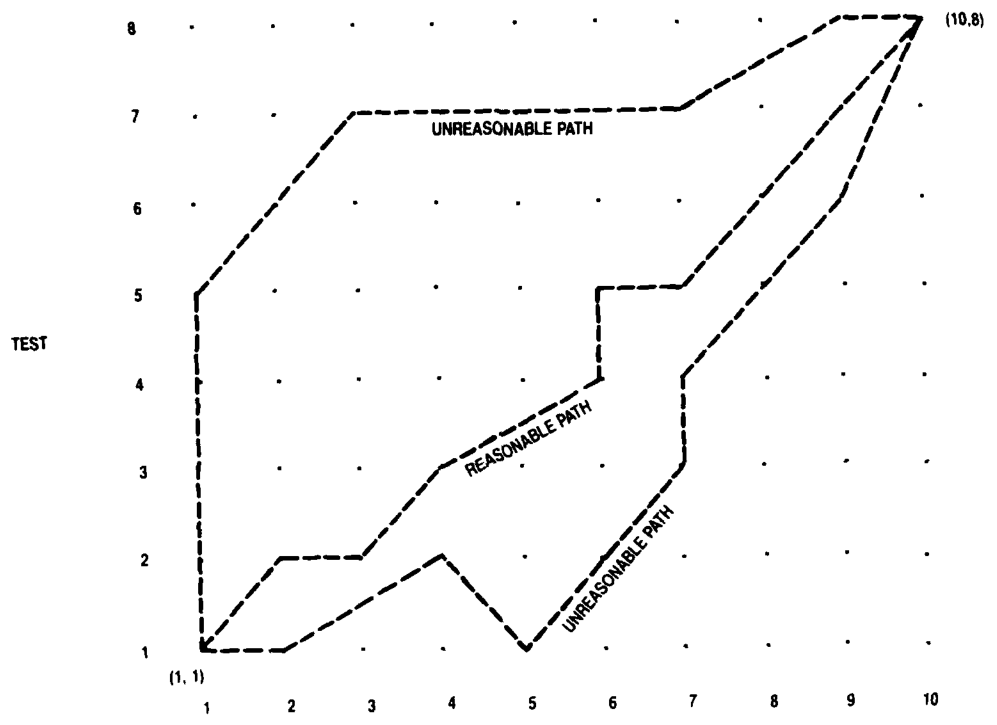


Figure 3.2 UNCONSTRAINED DTW GRID

k-th point: $(i(k),j(k))$

s-th intermediate point: $(i(k) - \sum_{\ell=1}^{s} a_\ell,\ j(k) - \sum_{\ell=1}^{s} b_\ell)$

(k - 1)st point: $(i(k-1),j(k-1)) = (i(k) - \sum_{\ell=1}^{L} a_\ell,\ j(k) - \sum_{\ell=1}^{L} b_\ell)$

This representation of local path constraints provides a great deal of flexibility in their choice. The left-hand side of figure 3.3 illustrates the Type 3 constraints of Myers et al. [11], which are specified by the four productions:

P1: (1,0)(1,1)
P2: (1,0)(1,2)
P3: (1,1)
P4: (1,2)

These four productions define four distinct possible local paths to a given point (i(k),j(k)) in the DTW grid, coming from the points (i(k) - 2, j(k) - 1), (i(k) - 2, j(k) - 2), (i(k) - 1, j(k) - 1), and (i(k) - 1, j(k) - 2), respectively. The first two of these local paths also pass through the intermediate point (i(k) - 1, j(k)). Note that for any local path to be valid, its starting point (i(k - 1), j(k - 1)) and its end point (i(k),j(k)) must belong to the valid point set.

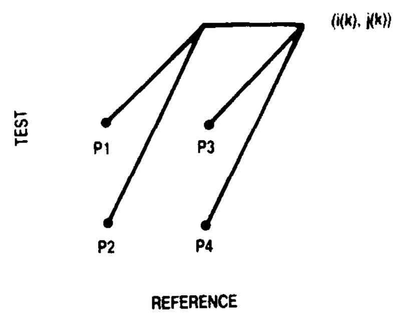
A zero value for an a (b) in a production implies that the corresponding reference (test) frame is to be matched with more than one test (reference) frame. A value greater than one, on the other hand, results in one or more reference (test) frames being skipped (not matched) altogether. Thus, paths P1 and P2 of the Type 3 constraints allow a given test frame to be matched with more than one reference frame, while paths P2 and P4 permit the skipping of a test frame. Under these constraints, each reference frame is used exactly once. Corresponding to the Type 3 constraints is a reflected version, the Type 3a constraints shown in the right-half of figure 3.3. These are specified by the four productions:

P1: (0,1)(1,1)
P2: (0,1)(2,1)
P3: (1,1)
P4: (2,1)

For these constraints, paths P1 and P2 match a given reference frame against more than one test frame, while paths P2 and P4 permit a reference frame to be skipped, but each test frame is used once and only once.

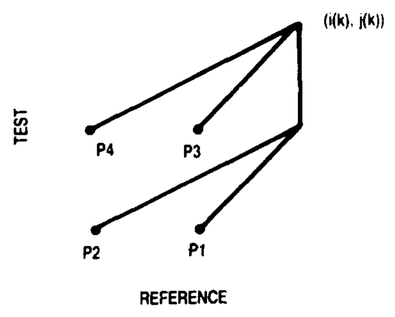
While there is no apparent reason for claiming that one set of constraints will perform better than the other, it seems more natural to require that each test frame be matched exactly once, while allowing reference frames to be skipped or used more than once. Thus, we tend to prefer the Type 3a constraints over the Type 3. Myers et al. in effect tested both types (along with a number of other sets of local constraints) by using the Type 3 constraints but allowing the assignment of test and reference to the x- and y-axes to be reversed. They found better results for the reversed case,

IA-72.528



P1: (1, 0) (1, 1)
P2: (1, 0) (1, 2)
P3: (1, 1)
P4: (1, 2)

TYPE 3



P1: (0, 1) (1, 1)
P2: (0, 1) (2, 1)
P3: (1, 1)
P4: (2, 1)

TYPE 3a

Figure 3.3 LOCAL PATH CONSTRAINTS

which corresponds to using the Type 3a constraints. Furthermore, there are computational advantages to requiring that each test frame be matched exactly once; for then, the logarithm of the squared gain, σ_f^2, of the test power spectrum does not need to be evaluated in determining the local distortion values. This is because each calculated value of σ_f^2 will be used exactly once in any legal path from (1,1) to (m,n), and therefore can have no influence upon determining the best path.

Associated with each local path to a grid point (i,j) is a path cost which is a weighted sum of the local distortion values for grid points passed through by the path. One of the simplest of weight functions takes the form

$$w(k) = i(k) - i(k - 1) \qquad (3.2)$$

For this weight function, the weight assigned to a local path is the distance traversed in the reference direction (i.e., the sum of the a´s in the production defining the local path). It is customary to divide the weight equally among the segments forming the path. Thus, for type 3 local constraints, this weight function assigns unit weights to all path segments, whereas for the type 3a constraints a fractional weight will result for the segments of path P1.

Local constraints limit the valid point set making up the DTW grid in the following manner. For each procedure P of a local constraint, let sum(a) denote the sum of all the a´s and let sum(b) denote the sum of all the b´s. The slope of the local path is given by the ratio sum(b)/sum(a). Let emax and emin denote the maximum and minimum slopes, respectively, obtained over all productions P comprising the local constraint. If we draw lines of slope emin and emax through the endpoints (1,1) and (m,n), the resulting four lines define a parallelogram in the initial DTW grid within which all valid points must lie (see figure 3.4). Points intermediate to local paths may lie outside this parallelogram, but the endpoints of such paths must themselves lie on or within the parallelogram. In figure 3.4 the parallelogram resulting from the Type 3 constraints of [11] is shown, drawn in solid lines, for an illustrative example representing ten reference frames and eight test frames.

Global path constraints were introduced by Sakoe and Chiba [4] to further delimit the legal point set. These constraints take the form

$$|i(k) - j(k)| \leq g \tag{3.3}$$

for some nonnegative integer g. They constrain the DTW path to lie within a corridor of width 2g centered on a 45-degree diagonal through the point (1,1). Of course, if $|m - n| > g$, then the end-point (m,n) cannot satisfy the global constraint, and no legal DTW path can be found. Thus, in addition to restricting where the path can lie, the global constraint can be used to rule out altogether a search for the shortest path whenever the lengths of the test and reference utterances are too dissimilar.

A choice of $g = 0$ will permit no path unless $n = m$, in which case all local paths must begin and end on the diagonal from (1,1) to (m,m). The global constraint usually limits the DTW grid by cutting off the interior corners of the parallelogram defined by the local constraints. In the example illustrated in figure 3.4, only the lower right corner is in fact cut off by the severe global constraint $g = 2$. The resulting legal points comprising the DTW grid are shown as solid grid points. The hollow or empty points lying outside the parallelogram are intermediate points which may be passed through in traversing certain local paths which begin and end in the legal point set. The selected local distortion measure must be evaluated for such intermediate points as well as for the points in the legal set.

3.2.2 DTW Path Computations

Dynamic time-warping for speech recognition was first formulated as a problem in dynamic programming by Sakoe and Chiba [4]. In fact, however, the problem of finding the best path through the DTW grid reduces to a special simple case of dynamic programming known as the shortest route problem. This problem can be stated briefly as follows: Given a connected graph with two distinguished nodes A and B and with a cost associated with each arc from a node i to a node j of the graph, find the path (i.e., sequence of arcs) from A to B whose summed cost is a minimum. Algorithms for finding an optimal solution to this problem were first given (independently) by Moore [13] and Dantzig [14]. Subsequently, Bellman [15] formulated the shortest route problem as a dynamic programming problem.

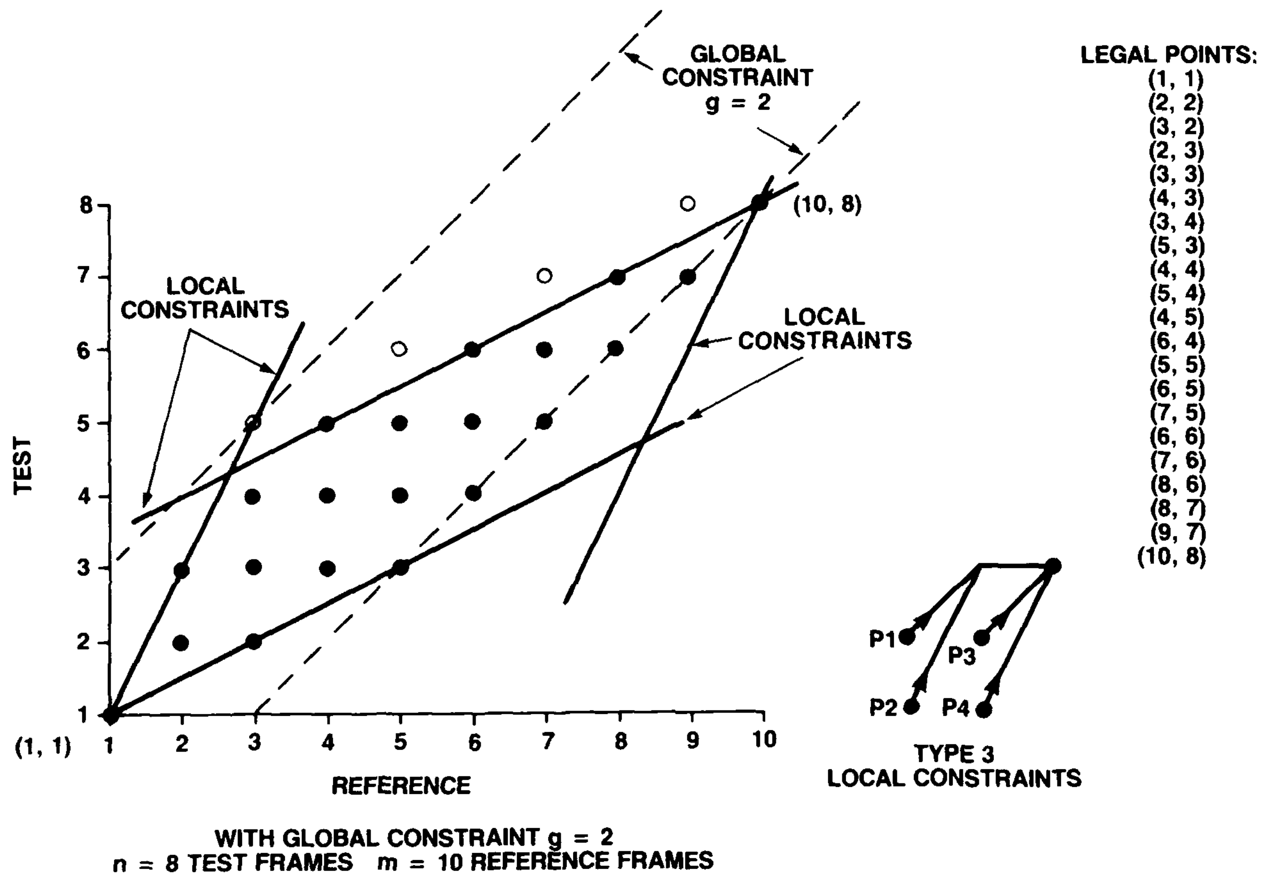


Figure 3.4 GRID FOR TYPE 3 LOCAL CONSTRAINTS

The network, or graph, to which the shortest route algorithm is applied is defined as follows: Nodes of the graph correspond to legal points of the DTW grid, with the grid point (1,1) as the node A and the grid point (m,n) as the node B. The arc costs are defined as weighted sums of local distortions obtained for matches of reference and test frames corresponding to grid points passed through in going from the grid point associated with node i to that associated with node j. For the type 3 local constraints and the weight function defined in equation (3.2), the costs defined for arcs of the network derived from the DTW grid have the form

$$\begin{aligned} c(P1) &= c(P2) = d_{i-1,j} + d_{ij} \\ c(P3) &= c(P4) = d_{ij} \end{aligned} \tag{3.4}$$

where d_{ij} is the local distortion calculated between the ith reference frame and the jth test frame. The network derived from the DTW grid for the example given previously in figure 3.4 and assuming weight function (3.2) is shown in figure 3.5.

The minimum cost c_{ij} for any path to the node (i,j) is computed (under type 3 constraints and weight function (3.2)) as

$$\begin{aligned} c_{ij} = \text{Min} \big(& d_{ij} + c_{i-1,j-1},\ d_{ij} + c_{i-1,j-2}, \\ & d_{ij} + d_{i-1,j} + c_{i-2,j-1},\ d_{ij} + d_{i-1,j} + c_{i-2,j-2} \big) \end{aligned} \tag{3.5}$$

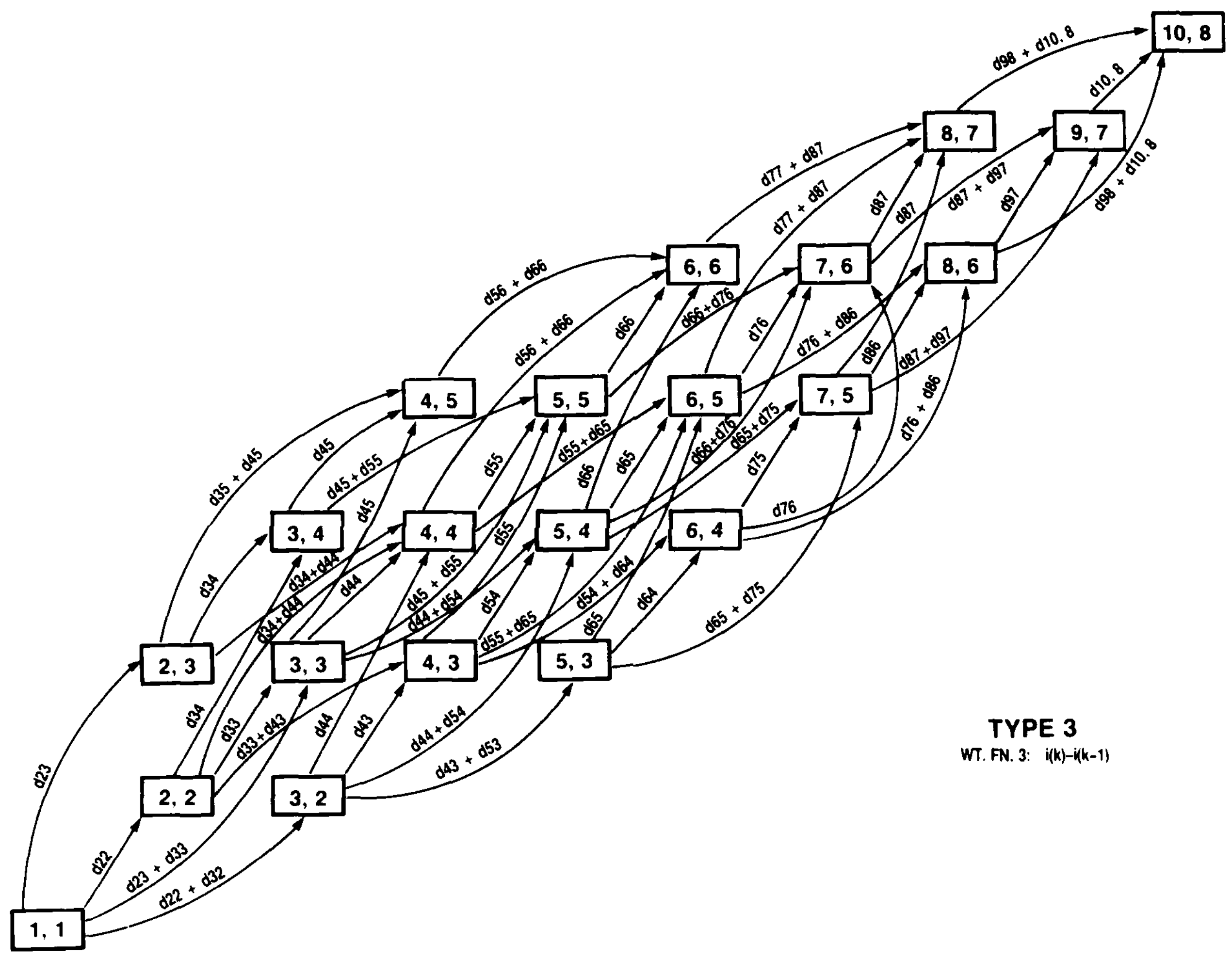


Figure 3.5 DTW NETWORK

where the first two terms of the minimization are the cost of reaching (i,j) by local paths P3 and P4, and the latter two terms are the cost of reaching (i,j) by local paths P1 and P2. Let

$$\hat{c}_{ij} = d_{ij} + \text{Min}\left(c_{i-1,j-1}, c_{i-1,j-2}\right). \qquad (3.6)$$

Then

$$\hat{c}_{i-1,j} = d_{i-1,j} + \text{Min}\left(c_{i-2,j-1}, c_{i-2,j-2}\right) \qquad (3.7)$$

and

$$c_{ij} = \text{Min}\left(\hat{c}_{ij}, d_{ij} + \hat{c}_{i-1,j}\right). \qquad (3.8)$$

c_{mn}, the minimum cost for any path to node (m,n), is the score returned by the DTW algorithm.

The shortest path computation for type 3 local constraints and weight function (3.2) can be summarized as follows:

1. Compute the local distortion d_{ij} from the test frame correlation coefficients $r_n(j)$ and the reference frame inverse correlation coefficients $u_n(i)$

2. Compute $\hat{c}_{ij} = d_{ij} + \mathrm{Min}\left(c_{i-1,j-1},\ c_{i-1,j-2}\right)$

3. Compute $c_{ij} = \mathrm{Min}\left(\hat{c}_{ij},\ d_{ij} + \hat{c}_{i-1,j}\right)$

We would like to employ RNS for step 1, the computationally most intensive calculation in a DTW-based speech recognition system. The problem which arises if RNS is used is the magnitude comparisons required for steps 2 and 3.

3.3 RNS IMPLEMENTATION OF THE DYNAMIC TIME-WARPING ALGORITHM

In order to make use of RNS for the local distortion calculations of a DTW algorithm, it is highly desirable to remain within RNS for the entire DTW shortest path computation, leaving only to convert the final score output by the algorithm for thresholding and comparison with other scores to select the best match. As we have seen in section 3.2, solution of the shortest path problem involves a sequence of additions and magnitude comparisons. In general, magnitude comparisons cannot be efficiently performed within RNS. However, the magnitudes being compared in the shortest path computation may be similar. If their difference in absolute value does not exceed half the largest modulus in use, then relative magnitude can be determined without leaving RNS, simply by testing the difference modulo this largest modulus. As a first approach to an RNS implementation we tested the following hypothesis:

SHORTEST PATH DISTANCE HYPOTHESIS

The absolute differences of cumulative distances compared in steps 2 and 3 of the shortest path *algorithm will generally not* exceed half the largest modulus we are willing to employ in a residue number system of practical size.

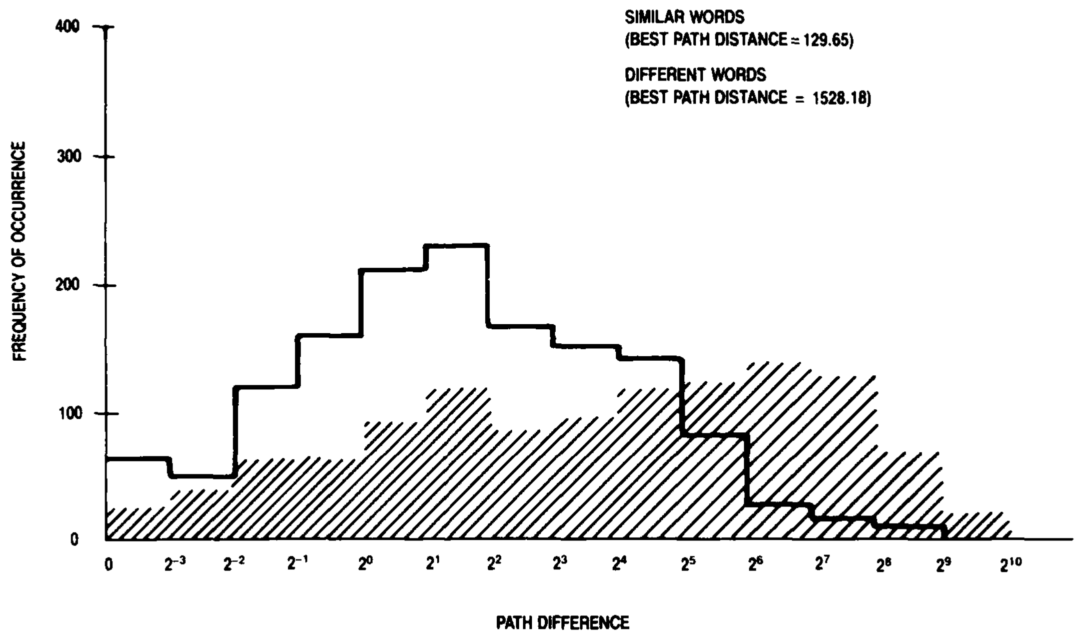
This hypothesis was tested by carrying out simulations using our DTW simulation program. Histograms of differences arising in the shortest path calculations were generated. The magnitudes of these differences depend upon the choice of local distortion measure. We looked at what these differences typically are in a conventional implementation for each of three distortion measures provided for in the simulation program: Itakura-Saito distortion, Gain-Optimized Itakura-Saito distortion, and Gain-Normalized Itakura-Saito distortion [3]. Three cases were examined:

1) identical test and reference templates;
2) similar test and reference templates; and
3) different test and reference templates.

The second case arises when the test and reference utterances are different productions of the same word; the third arises when the test and reference utterances are productions of different words.

Results are summarized in Table 3.1, which gives the smallest and largest differences encountered in each of the three cases for each of the three distortion functions, together with the number of divisions required to give a reasonable portrayal of the histogram. The width of each division, except the first, is half that of its successor. The first division has the same width as its successor in order that all differences may be counted with a reasonable number of divisions. The distributions for the Itakura-Saito metric are plotted in figure 3.6 for the cases of similar words (clear) and different words (cross-hatched).

IA-72.530



ITAKURA-SAITO METRIC

Figure 3.6 DISTRIBUTION OF PATH METRIC DIFFERENCES

Table 3.1

Differences Arising in the Shortest Path Calculations of the Dynamic Time-Warping Algorithm

Distortion Measure	Templates	Smallest	Largest	Divisions
	identical	.0006	374.9607	13
I-S	similar	.0031	352.5326	13
	different	.0017	733.7631	14
	identical	.0004	11.8593	11
G-O	similar	.0002	7.3737	10
	different	.0007	11.2536	11
	identical	.0009	37.7742	13
G-N	similar	.0011	23.4734	12
	different	.0027	151.2043	14

The number of divisions needed to cover the range of differences was at least ten for all distortion measures employed. It was concluded that our first shortest path distance hypothesis is not valid for distortions based on the Itakura-Saito distance measure. Therefore, we considered quantization of the distortion function leading to a modified distance hypothesis.

SECOND DISTANCE HYPOTHESIS

With appropriate quantization of local distortion values, the absolute differences of cumulative distances arising in steps 2 and 3 will not generally exceed half the largest modulus we are willing to employ in a residue number system of practical size.

For testing this revised distance hypothesis, an extreme quantization of distortion function values to a single bit (0 = match, 1 = no-match) was employed. A breakpoint threshold was chosen; all distortion values exceeding the threshold were then set equal to 1; all less than or equal to the threshold were set equal to 0. To select the breakpoint, we first compiled histograms of distortion function values for matching similar test and reference templates (different productions of the same word) and different templates (different words). These are shown (for the Itakura-Saito metric) in figure 3.7, where, as before, the histogram for different templates is cross-hatched. We seek a breakpoint that discriminates well between the two different comparisons. On the basis of this study, a breakpoint threshold of .5 was selected.

Histograms were then compiled of all finite differences, in absolute value, arising in the DTW shortest path calculations, by running many test patterns against the entire reference library. Under the extreme quantization employed, most of these differences became zero (table 3.2); the nonzero differences were both relatively few in number and small in size. This result supports the second hypothesis that, with appropriate quantization of local distortion values, it is feasible to carry out the DTW calculations in the shortest route algorithm in a residue number system of practical size.

IA-72,531

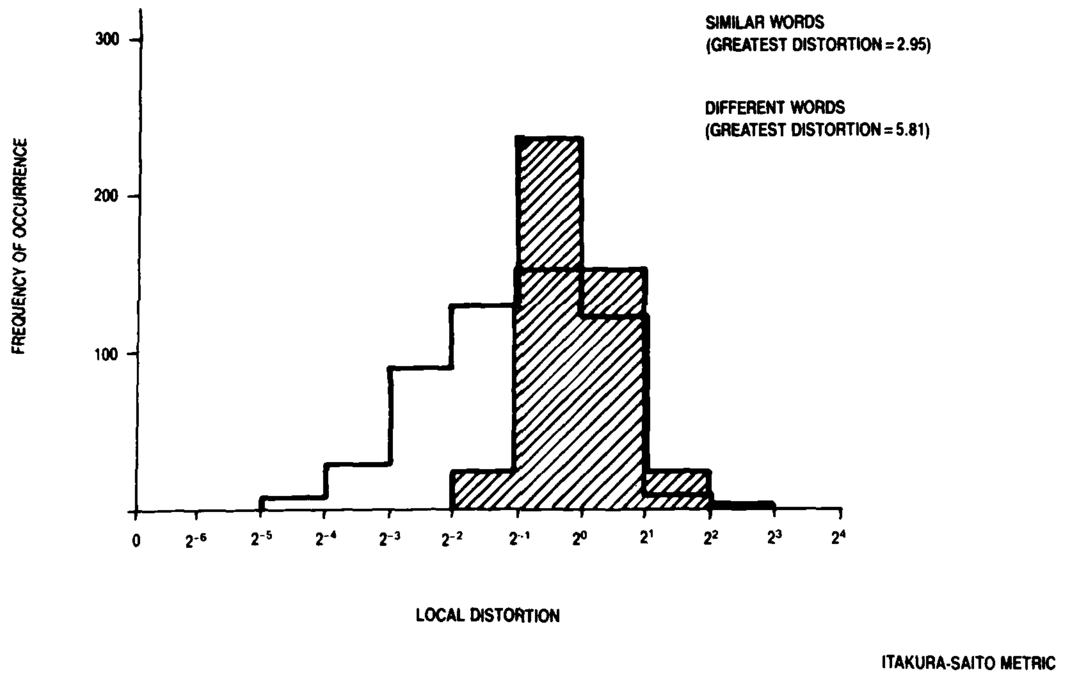


Figure 3.7 DISTRIBUTION OF LOCAL DISTORTION VALUES

Table 3.2

Histogram of Differences in DTW Shortest Path Calculations: One-bit Quantization of Gain-Normalized Itakura-Saito Distortion Function Values - Breakpoint = .5

Difference	Frequency of Occurrence
0	2227162
1	152344
2	108417
3	73399
4	28900
5	17500
6	9726
7	5321
8	3654
9	1918
10	1149
11	772
12	510
13	300
14	141
15	94
16	59
17	35
18	18
19	22
20	5
21	3
22	7
23	2
24	2
25	5
26	2
27	2

Quantization within RNS is difficult, in effect calling for sign determinations. Our solution has been to employ a two-part quantization, first quantizing from the range of the RNS into the range of a single modulus, and then requantizing to a single bit, using a threshold. The method developed for quantizing to the range of a single modulus will be described in section 3.4.

The revised shortest path computation is as follows:

1. Compute d_{ij} from $r_n(j)$ and $u_n(i)$

2. Quantize to single bit d'_{ij} : 0 = match, 1 = no-match

3. Compute $\hat{c}_{ij} = d'_{ij} + \mathrm{Min}(c_{i-1,j-1}, c_{i-1,j-2})$

4. Compute $c_{ij} = \mathrm{Min}(\hat{c}_{ij}, d'_{ij} + \hat{c}_{i-1,j})$

Figure 3.8 is a block diagram of an RNS implementation of a DTW-based speech recognition system. After detection of a test utterance, input values, possibly scaled, are converted to RNS for calculation of the test correlation coefficients. Inverse correlation coefficients from the reference library are scaled and converted to RNS (this would normally be done before storing them in the library), and the distortion function is computed as an inner product of vectors in RNS. The output of this calculation is quantized in two steps, first to the range of a single modulus, and then to a single bit. The shortest path through the DTW grid is obtained in an RNS of reduced size as described in section 3.5. The resulting

IA-72,532

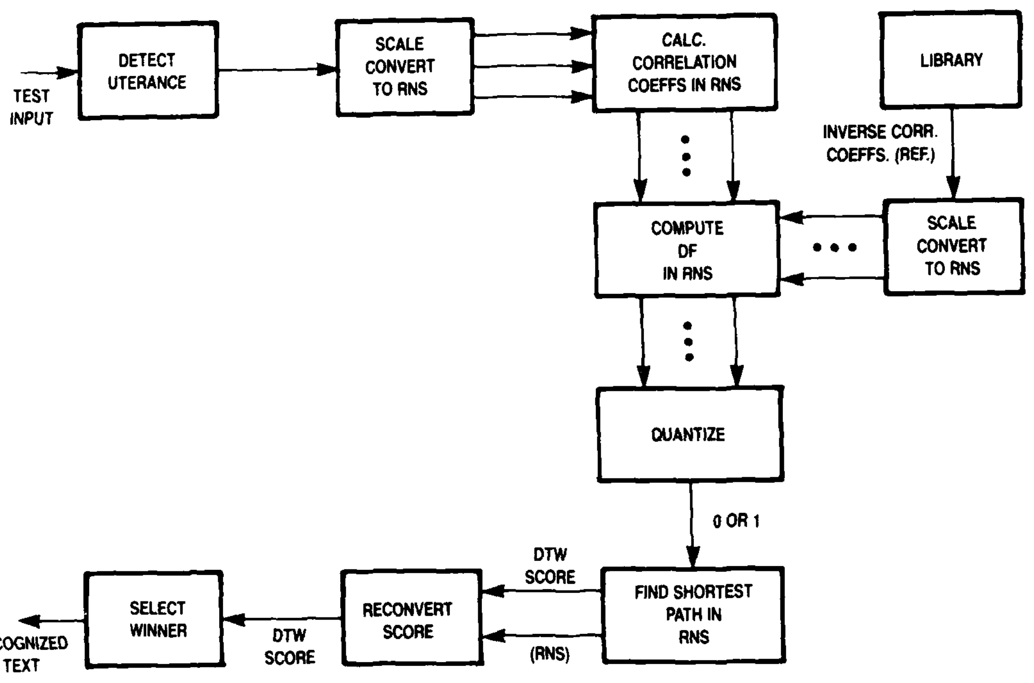


Figure 3.8 RNS IMPLEMENTATION OF DTW ALGORITHM

score is reconverted to a conventional number system for thresholding and selection of the winning text. In section 3.6 questions of RNS range and scaling are discussed briefly. Simulation results for an RNS implementation of the SRS of figure 3.8 are contained in section 3.7.

3.4 QUANTIZATION IN A RESIDUE NUMBER SYSTEM

3.4.1 Introduction

Our simulation experiments support the hypothesis that, with appropriate quantization of local distortion values, it is feasible to perform the DTW shortest path calculations within a practical-sized residue number system (RNS). However, this raises the question as to how an appropriate quantization of these values is to be obtained without first leaving RNS. We propose a two-phase quantization, first from the range of the RNS to the range of a single modulus, and then to a single bit. In this section, we show how to perform the first quantization in RNS.

At first glance, quantization of values within RNS appears to be a formidable problem requiring, in effect, a series of magnitude comparisons or sign determinations. In fact, however, the problem may be greatly simplified because, at least in some applications, there is some tolerance for error. Quantization divides the range of an observed variable into, say, k intervals, and replaces each value by the index of the interval within which it lies. The breakpoint dividing interval i from interval i + 1 is somewhat arbitrary, and it is expected that, at least in our speech recognition application, very little harm will come from errors made in the neighborhood of the breakpoint which cause a value lying in the i-th interval to be erroneously recorded as lying in the (i + 1)-st interval or vice versa. Based upon this tolerance of errors in the breakpoint neighborhood, we have devised a method for quantizing the values in the range of an RNS into the range of a single modulus m_i, in effect scaling by $\hat{m}_i = M/m_i$, where $M = m_1 m_2 \ldots m_n$ is the product of the n moduli m_i, assumed to be relatively prime in pairs, which comprise the RNS. The method can be employed for all sets of relatively prime moduli, but involves some calculation.

The remainder of this section is divided into three parts dealing, respectively, with the quantization function t(x) , the calculations required to evaluate t(x), and the choice of moduli.

3.4.2 The Quantization Function t(x)

Any integer x can be represented in a residue number system by

$$x = \sum_{i=1}^{n} \hat{m}_i \, |x/\hat{m}_i|_{m_i} - MA(x) \qquad (3.9)$$

where $|x|_m = x - m[x/m]$, [y] denotes the greatest integer contained in y, and A(x) is an integer-valued function first studied by Aiken and Semon [16] and whose range is discussed in [17], Appendix A. Suppose we divide our moduli m_i into two groups which we call p´s and q´s, where $p_1 = m_1$, $p_2 = m_2$, $q_1 = m_3$, $q_2 = m_4$, ..., $q_{n-2} = m_n$, and define $P = p_1 p_2$, $Q = q_1 q_2 \cdots q_{n-2}$, and $\hat{q}_i = Q/q_i$. Consider now the two RNS defined by the p´s and q´s. We can express x in the first system as

$$x = p_2 |x/p_2|_{p_1} + p_1 |x/p_1|_{p_2} - PA_P(x) \qquad (3.10)$$

and $-x$ in the second system as

$$-x = \sum_{i=1}^{n-2} \hat{q}_i |-x/\hat{q}_i|_{q_i} - QA_Q(-x). \qquad (3.11)$$

Let $\sigma(x) = p_2 |x/p_2|_{p_1} + p_1 |x/p_1|_{p_2} + \sum_{i=1}^{n-2} \hat{q}_i |-x/\hat{q}_i|_{q_i}.$ (3.12)

Then, by adding (3.10) and (3.11)

$$\sigma(x) = PA_P(x) + QA_Q(-x) \qquad (3.13)$$

Define $\bar{Q} = |-1/Q|_P$ (3.14)

$$s_1(x) = |\bar{Q}\sigma(x)|_{p_1} = |-A_Q(-x)|_{p_1} \qquad (3.15)$$

and

$$s_2(x) = |\bar{Q}\sigma(x)|_{p_2} = |-A_Q(-x)|_{p_2}. \qquad (3.16)$$

Note that s_1 and s_2 can be calculated within the RNS. Finally, let

$$t(x) = |s_2(x) - s_1(x)|_{p_1}. \tag{3.17}$$

What will the function $t(x)$ look like?

First consider the function $A_Q(-x)$. We have $A(kM + x) = A(x) - k$ (see reference [18]), and for $0 \leq x < M$ we have $0 \leq A(x) < n$, the number of moduli (see reference [17], Appendix A). Suppose first that the q-set consists of a single modulus. Then $-A_Q(-x) = -1$ for $0 < x \leq Q$, $-A_Q(-x) = -2$ for $Q < x \leq 2Q$, and so forth. The quantities $s_1(x)$ and $s_2(x)$ remain constant over any span $kQ < x \leq (k + 1)Q$; both decrease by one unit at the transition of x from the value $(k + 1)Q$ to $(k + 1)Q + 1$. However, their difference, modulo p_1, does not change *except at* every p_2-th such transition. The first such change occurs for $x = p_2 \cdot Q + 1$, when $-A_Q(-x)$ becomes $-(p_2 + 1)$. A similar change will occur every p_2Q values thereafter. Thus, when the q-set consists of a single modulus, $t(x)$ takes on the p_1 values $0, 1, \ldots, p_1 - 1$ in some order in blocks of length $p_2 \cdot Q = \hat{m}_1$.

Next, consider the case where the q's consist of two moduli, q_1 and q_2. Then $-A_Q(-x)$ takes on the two values -1 and -2 in the range $0 < x \leq Q$, the two values -2 and -3 in the range $Q < x \leq 2Q$, and so forth. The quantities s_1 and s_2 can vary by one unit over the range $kQ \leq x < (k + 1)Q$, but their difference is normally constant modulo p_1. Both decrease by one unit at the transition points but, again, the difference modulo p_1 is unchanged except at every p_2-th transition, and at certain values in the last sub-block of length Q before such a transition, namely, those values

$x + (kp_2 - 1)Q$ for $0 < x \leq Q$ for which $A_Q(-x) = 2$ or, equivalently, those values for $0 \leq x < Q$ for which $A_Q(x) = 1$. For these values $t(x)$ turns too soon, creating an ambiguous neighborhood at the transition point. The length of this ambiguous neighborhood is known [17]. Let j be the unique integer in $[0, Q - 1]$ satisfying $|-j/\hat{q}_i|_{q_i} = 1$ for $i = 1, \ldots, n - 2$. Then the ambiguous neighborhood has size $j + 1$.

<u>Example 1</u>: $p_1 = 4$, $p_2 = 9$, $q_1 = 5$, $q_2 = 7$

Then $j = 23$ and we should find four blocks of length 315, with an ambiguous area of length 24 at the end of each block. See table 3.3.

Table 3.3

$t(x)$ for 4, 9, 5, 7 Residue Number System

x	t(x)
1 - 291	1
292 - 315	1 or 2
316 - 606	2
607 - 630	2 or 3
631 - 921	3
922 - 945	3 or 0
946 - 1236	0
1237 - 1260	0 or 1

The range of the system above 1236 should not be used, as errors which are made here would map a value which belongs in the last interval into the first interval. Other errors are presumed to be relatively harmless. The size of the ambiguous neighborhood depends only on the q's.

Finally, consider the case where there are more than two q's. In this case, $A_Q(x)$ may take on values greater than 1 in the interval $0 \leq x < Q$. If k is the largest value of $A_Q(x)$ in $[0, Q - 1]$, and j is defined as before, then the ambiguous neighborhood has size $(k - 1)Q + j$. The size attainable by k is treated further in [17].

Example 2: $p_1 = 7$, $p_2 = 11$, $q_1 = 2$, $q_2 = 3$, $q_3 = 5$

For this example $k = 1$ and $j = 29$. We expect blocks of length 330, with an ambiguous range of length 30 at the end of each block. See table 3.4

Table 3.4

t(x) for 7, 11, 2, 3, 5 Residue Number System

x	t(x)
1 - 300	4
301 - 330	4 or 1
331 - 630	1
631 - 660	1 or 5
661 - 960	5
961 - 990	5 or 2
991 - 1290	2
1291 - 1320	2 or 6
1321 - 1620	6
1621 - 1650	6 or 3
1651 - 1950	3
1951 - 1980	3 or 0
1981 - 2280	0
2281 - 2310	0 or 4

Example 2 (continued):

The range above 2280 should not be used, as an error occurring here would map a value belonging to the last interval into the first. Notice that the quantized values, t(x), of table 3.4 are inappropriately ordered.

Since from (3.12) $\sigma(0) = 0$ (cf. equations 3.15 to 3.17), we have $t(0) = 0$, and, since there is a transition when x passes from 0 to 1, $t(1) \neq 0$. Using this fact, we can transform the t(x) values into an ordered sequence $(1, 2, \ldots, p_1 - 1, 0)$ by a premultiplication by $|t(1)^{-1}|_{p_1}$. For example 2, we have $t(1) = 4$ and $|t(1)^{-1}|_{p_1} = |1/4|_7 = 2$, resulting in table 3.5 in place of table 3.4:

Table 3.5

$|t(1)^{-1} \cdot t(x)|_{p_1}$ for 7, 11, 2, 3, 5 RNS

x	$\lvert t(1)^{-1} \cdot t(x)\rvert_{p_1}$	$\lvert t(1)^{-1} \cdot t(x) - 1\rvert_{p_1}$
1 - 300	1	0
301 - 330	1 or 2	0 or 1
331 - 630	2	1
631 - 660	2 or 3	1 or 2
661 - 960	3	2
961 - 990	3 or 4	2 or 3
991 - 1290	4	3
1291 - 1320	4 or 5	3 or 4
1321 - 1620	5	4
1621 - 1650	5 or 6	4 or 5
1651 - 1950	6	5
1951 - 1980	6 or 0	5 or 6
1981 - 2280	0	6
2281 - 2310	0 or 1	6 or 0

The third column of table 3.5 shows the effect of further modifying the quantization function to eliminate the bias evident in the second column. Again, observe that the useful range of x extends only to 2280 rather than 2310.

3.4.3 Calculations

Calculation of $s_1(x)$ requires a simple table lookup for n - 1 quantities which are summed in a modulo p_1 adder; calculation of $s_2(x)$ requires a simple table lookup for n - 1 quantities which are summed in a modulo p_2 adder. Calculation of t(x) requires one additional summation (subtraction) in the modulo p_1 adder.

From (3.12) (3.14) and (3.15) we have

$$s_1(x) = |\bar{Q}\sigma(x)|_{p_1} = \Big| |\bar{Q}|_{p_1} |x|_{p_1} + \sum_{i=1}^{n-2} |\bar{Q}\hat{q}_i|_{p_1} |-x/\hat{q}_i|_{q_i} \Big|_{p_1}$$

$$= \Big| |\bar{Q}x|_{p_1} + \sum_{i=1}^{n-2} \big| |-1/q_i|_{p_1} |-x/\hat{q}_i|_{q_i} \big|_{p_1} \Big|_{p_1}.$$

Similarly,

$$s_2(x) = \Big| |\bar{Q}x|_{p_2} + \sum_{i=1}^{n-2} \big| |-1/q_i|_{p_2} |-x/\hat{q}_i|_{q_i} \big|_{p_2} \Big|_{p_2}. \tag{3.19}$$

The quantities

$$a_i = \big| |-1/q_i|_{p_1} |-x/\hat{q}_i|_{q_i} \big|_{p_1} \tag{3.20}$$

and

$$b_i = \left| \left| -1/q_i \right|_{p_2} \left| -x/\hat{q}_i \right|_{q_i} \right|_{p_2} \qquad (3.21)$$

can be precomputed for the q_i values $|x|_{q_i}$ and stored in (hard-wired) tables.

Example 3: $p_1 = 4$, $p_2 = 5$, $q_1 = 7$, $q_2 = 9$.

Then $\bar{Q} = 13$, $a_1 = ||3x|_7|_4$, $a_2 = |3|5x|_9|_4$, $b_1 = |2|3x|_7|_5$, $b_2 = ||5x|_9|_5$. The required lookup tables are illustrated in table 3.6

Table 3.6

Tables for Quantization in 4, 5, 7, 9 RNS

x	a_1	a_2	b_1	b_2	$\|3x\|_5$
0	0	0	0	0	0
1	3	3	1	0	3
2	2	3	2	1	1
3	2	2	4	1	4
4	1	2	0	2	2
5	1	1	2	2	-
6	0	1	3	3	-
7	-	0	-	3	-
8	-	0	-	4	-

The last column provides a table for $|3x|_5$, needed for computing $s_2(x)$. Since $|\bar{Q}|_4 = 1$, no table is needed in this example to obtain $|\bar{Q}x|_4 = |x|_4$. These tables should be easy to implement. For example, if $|x|_9$ is given in binary form, the b_2 value can be obtained in this example simply by dropping the last bit.

Figure 3.9 is a schematic block diagram depicting the calculation of $t(x)$ from the n residues $|x|_{p_1}$, $|x|_{p_2}$, $|x|_{q_1}$, ..., $|x|_{q_{n-2}}$. The quantities $|\bar{Q}x|_{p_1}$, $|\bar{Q}x|_{p_2}$, a_k, and b_k ($k = 1, \ldots, n - 2$) are obtained from the residues by hard-wired table lookups and fed to the two modular adders, producing the quantities $s_1(x)$ and $s_2(x)$, whose difference, modulo p_1, defines $t(x)$. The modular adders in figure 3.9 are assumed to be $(n - 1)$-input adders. If only two-input adders are available, the summations producing $t(x)$ require $n - 1$ stages, as follows:

Step 1: Add $|\bar{Q}x|_{p_1}$ and a_1 in the modulo p_1 adder; add $|\bar{Q}x|_{p_2}$ and b_1 in the modulo p_2 adder.

Step k: Add a_k to the contents of the modulo p_1 adder; add b_k to the contents of the modulo p_2 adder.

Step n-2: Add a_{n-2} to the contents of the modulo p_1 adder and output the result, $s_1(x)$; add b_{n-2} to the contents of the modulo p_2 adder and output the result, $s_2(x)$.

Step n-1: Subtract $s_1(x)$ from $s_2(x)$ in a modulo p_1 adder, producing the desired result $t(x)$.

IA-72.533

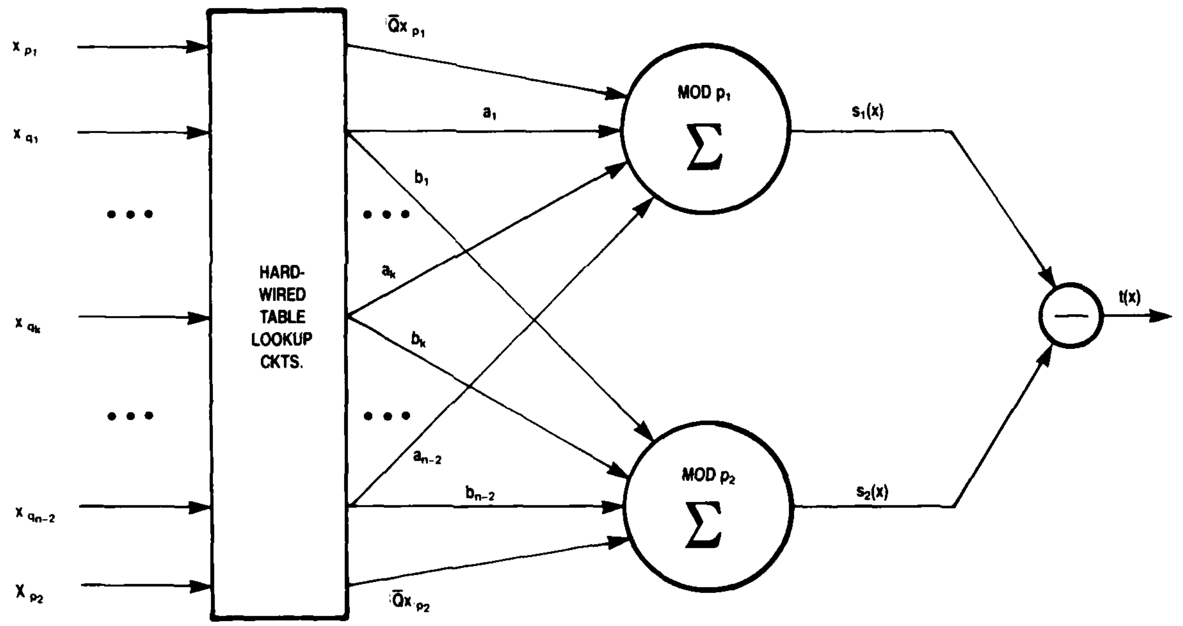
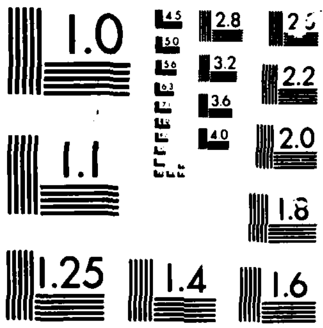


Figure 3.9 CALCULATION OF $t(x) = s_2(x) - s_1(x)|_{p_1}$

AD-A166 921 ADVANCED ARCHITECTURES FOR DIGITAL SIGNAL PROCESSORS 2/2
(U) MITRE CORP BEDFORD MA A L BEQUILLARD ET AL. OCT 85
MTR-9647 RADC-TR-85-203 F19628-84-C-0001
UNCLASSIFIED F/G 9/2 NL



MICROCOP' CHART

Quantization by $t(x)$ produces equally spaced intervals. If unequal intervals are desired, some additional computations may be required to map the $t(x)$ values into a reduced set. Modular reduction of $t(x)$ may also be required to obtain its modular representation, unless p_1 is the smallest of the n moduli m_i.

3.4.4. Choice of Moduli

From a given set of n relatively prime moduli m_i comprising an RNS, any given single modulus can be chosen as p_1, and any remaining modulus can be chosen as p_2. The designation of p_1 and p_2 determines the size of the modular adders required in the computation of $t(x)$ but, since these adders are required in any event, it should not influence the choice of p_1 and p_2. The selection of p_1 is primarily influenced by the particular quantization desired; the larger p_1 is, the more flexible are the choices for the ultimate quantization. On the other hand, if p_1 is the smallest of the n moduli, it may be possible to avoid modular reduction of the quantized values.

The size of the ambiguous range in each block is determined by the choice of the q s. It is less than (but of the order of) $(n - 3)Q$, whereas the block size itself is p_2Q. The larger the value of p_2, the greater the error-free portion of each block. It may, therefore, be desirable to choose p_2 to be the largest of the n moduli, but this choice is not critical. However, p_2 should probably be somewhat larger than $n - 3$.

3.4.5. Conclusion

A method has been gi"en for providing quantization within a residue number system from values (almost) anywhere in the range of the system into the range of a single modulus. An alternative, of course, is to perform a translation into a mixed radix number system of each value x to be quantized. This calls for a roughly equivalent amount of work (n - 1 table lookups and subtractions), but does not in itself produce the desired quantization, requiring a further division (or equivalent operation) upon the translated value, and a possible reconversion of the result to RNS. It is felt that there is a definite advantage to remaining within RNS for this computation. This rather gross quantization appears to be appropriate for mapping the results of local distortion computations into a small range suitable for performing the DTW shortest path calculations within RNS.

The method can easily be extended with the addition of more modular adders to allow quantization to the range of a subset of the moduli rather than to the range of a single modulus. To map values from range M to range $\hat{p}_k = p_1, p_2, \ldots, p_{k-1}$, where our RNS moduli have been divided into two sets with k and n-k members, respectively, we replace (3.10), (3.11), and (3.12) by

$$x = \sum_{i=1}^{k} \hat{p}_i |x/\hat{p}_i|_{p_i} - PA_P(x) \tag{3.22}$$

$$-x = \sum_{i=1}^{n-k} \hat{q}_i |-x/\hat{q}_i|_{q_i} - QA_Q(-x) \tag{3.23}$$

$$\sigma(x) = \sum_{i=1}^{k} \hat{p}_i |x/\hat{p}_i|_{p_i} + \sum_{i=1}^{n-k} \hat{q}_i |-x/\hat{q}_i|_{q_i} \tag{3.24}$$

and replace p_1 and p_2 by $\hat{p}_k$ and p_k, respectively, in equations (3.15), (3.16), and (3.17). To calculate $s_1(x)$ and $t(x)$ in RNS, we simply get a modular representation in terms of the p_i. Thus, we use k-1 adders to calculate $s_1(x)$ and $t(x)$, and one more to calculate $s_2(x)$.

3.5 RNS IMPLEMENTATION OF THE SHORTEST PATH ALGORITHM

Although a somewhat large range is required for an RNS implementation of the Itakura-Saito distortion function calculation (as will be seen from the discussion in section 3.6), the shortest path calculation itself can be successfully performed in an RNS of much smaller range. Two types of potential overflow must be considered. First, the magnitude comparisons of steps 3 and 4 of the revised shortest path computation of section 3.3 are to be performed in the largest residue channel. An overflow will result if the difference, in absolute value, between the cumulative path distances under comparison exceeds half the largest modulus employed. Second, the cumulative distances themselves are represented by residues in all channels used. An overflow results if the cumulative distance for the presumed best path exceeds the range of the RNS. This is a serious error, generally leading to a recognition error, for the resulting path score will be much lower than its true value.

We consider first this latter overflow possibility. Under weight function 3 of equation (3.2), the cumulative cost for any path cannot exceed the number of reference frames. While alternative weight functions can give higher costs, the most expensive weight function we have employed results in paths whose cumulative cost cannot exceed the sum of the number of test frames and the number of reference frames. Since the longest utterance in our test library consists of 72 frames, an RNS of range 144 or greater suffices for the shortest path computation (under one-bit quantization of distortion values). In a typical simulation run of sixty test cases against the entire library, the largest path score produced was 46, resulting from matching a 72-frame "four" and a 70-frame "five."

We plan to employ the two largest moduli from the set used for the distortion calculations as an RNS for the shortest path calculations. For most of our simulations this has been the pair of primes 73 and 71, providing a range much greater than needed for this computation.

The first type of overflow, though less harmful, is much more likely to occur, and care should be taken to ensure that the first residue channel is of sufficient size. Table 3.7 shows the number of occurrences of errors of the first type for various choices of RNS2 for the DTW path computations. In all cases shown, the correlation and Itakura-Saito distortion function computations were performed using an RNS1 composed of the five prime moduli {73,71,67,61,59}, and quantization of the distortion values was performed in two stages, first to the range (0,72), and then to a single bit (match or no-match) using a threshold value T = 16. The last column of the table shows the recognition error rate for simulations performed using our sixty-word test set consisting of different productions of the ten digits and "oh" (The setting of the global constraint always caused one error (1.7 percent) among the sixty test cases).

For the last two RNS {7,5} and {5,3} the RNS2 range is exceeded much of the time by the cumulative distances. For the remaining cases, the range is never exceeded by the cumulative distances. The results displayed in table 3.7 support the hypothesis that little harm results from occasionally overflowing the largest modulus in the path comparisons, provided that the number of overflows is not excessive. No degradation in recognition performance was observed until the largest modulus was reduced to 11, when the number of overflows exceeded 12000. No overflows occur when the largest modulus is 25 or greater.

Table 3.7

Number of Overflows of First Modulus and Recognition Error Rates for Various RNS2 Choices for DTW Calculations - One-Bit Quantization of Distortion Values - RNS1 = {73,71,67,61,59}; Threshold = 16

Moduli	Number of Overflows	Recognition Error Rate
25,23	0	1.7
23,21	8	1.7
21,19	44	1.7
19,17	314	1.7
17,15	364	1.7
15,13	976	1.7
13,11	4557	1.7
11,9	12390	6.7
9,7	38323	25.0
7,5	110438	61.7
5,3	294523	95.0

3.6 RNS RANGE AND SCALING

The Itakura-Saito distortion measure calculation has the form

$$d_{IS} = \text{scalar} + \sum_{n} r_n u_n \tag{3.25}$$

The range of the RNS to be employed must be sufficient to contain the end-product of this calculation. Occasional overflow of the RNS by the calculated distortion function would probably not cause much harm; frequent overflow could affect recognition accuracy adversely; constant overflow would destroy the information contained in the distortion measure and render it useless for speech recognition.

We assume that the test correlation coefficients r_n are computed in RNS. It is possible to perform some scaling down of the test utterance sample values before calculation of the r_n. The reference inverse correlation coefficients are normally small fractional values, and must be scaled up before conversion to integer values and RNS representation. Since this conversion will entail truncation error, it must be performed with some care.

If the test input sample values are scaled by the factor 2^{-k}, then the r_n values are scaled by 2^{-2k}. If then the reference inverse correlation coefficients are scaled up by a factor 2^h, the "scalar" must be scaled up by a factor 2^{h-2k}. The end result is that the distortion function is scaled up by 2^{h-2k}. For example, if we scale the test input sample values by 2^{-2} and the reference inverse correlation coefficients by 2^{31}, the result is to scale the distortion function by 2^{27}. Since we have seen (figure 3.7) unscaled Itakura-Saito distortions of the order of 2^3, we would expect to need an RNS range of about 2^{30} to contain this calculation.

In the RNS simulations reported in section 3.7, twelve-bit input sample values were used for the test utterances. After windowing (Hamming window) and RMS normalization, these were reconverted to integer values and clipped at ±2047. Before conversion to RNS, these values can be scaled down a little, but not much or they will incur truncation errors large relative to their size.

We looked at the unscaled inverse correlation coefficients. The largest values (for 12-bit inputs) tend to lie in the range 10^{-6} to 10^{-5}. These must be scaled up before conversion to integers and RNS. We would probably like to scale these up by something like 10^{8} to 10^{9} to make them comparable in size to the unscaled test correlation coefficients, but must take care that the resulting scaled distortion values not exceed the RNS range more than occasionally. In the next section, we show simulation results employing various scalings for the five-modulus RNS (73, 71, 67, 61, 59), with range approximately 1.25×10^{9}, or about 1.16×2^{30}.

3.7 SIMULATION RESULTS FOR IMPLEMENTATION OF DTW ALGORITHM IN RNS

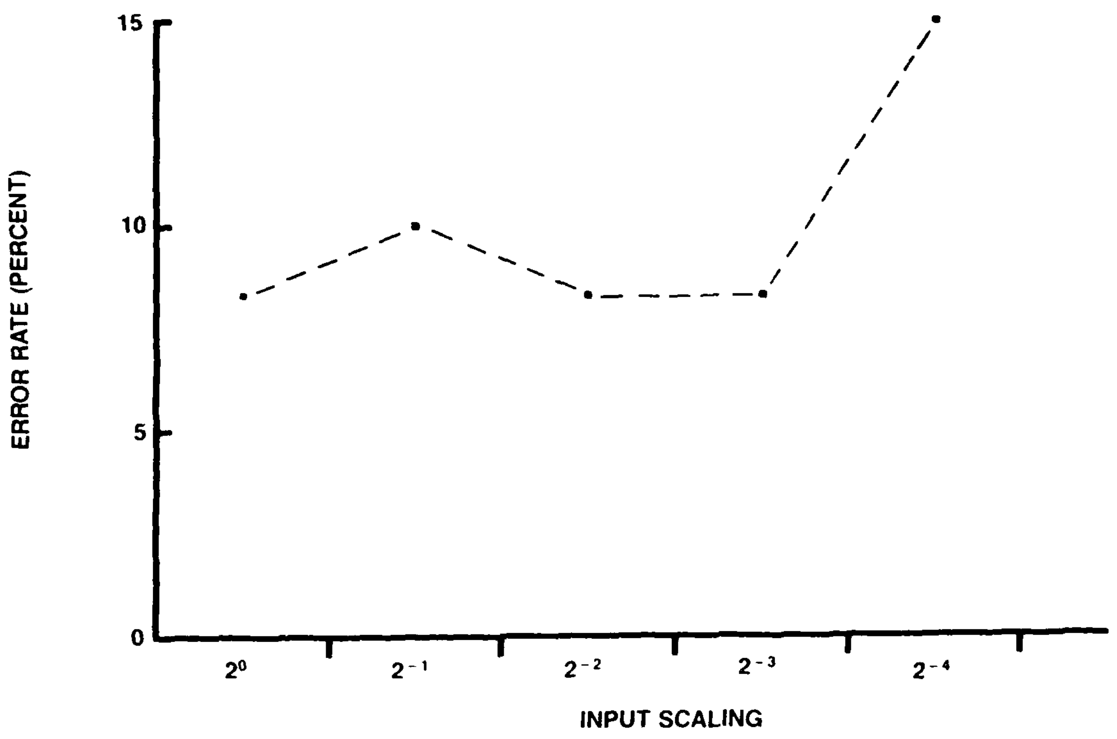
3.7.1 Simulation Test Set

In this section we describe the results of simulations to analyze the performance of an RNS-based DTW speech recognition system, such as that illustrated in figure 3.8. In all simulations, the Itakura-Saito distortion metric was used. All simulations were performed using a limited-size single-speaker library consisting of sixty utterances from an eleven-word vocabulary (the ten digits 0 - 9 and "oh"). The same set of sixty utterances was used as the test set. Scoring a success required that both the best and second best guesses be the correct text. Tie scores, which would result in ambiguity, were counted as failures. The sixty test cases always resulted in at least one failure which was caused by the setting of the global constraint and not by the RNS implementation or the quantization of Itakura-Saito distortion values. A conventional implementation, running with full range distortion values, would have made the same error, as the global constraint eliminated from consideration all reference patterns, other than that identical to the test, with same text value in this one case. For most of the simulations the RNS (73, 71, 67, 61, 59), with range approximately 2^{30}, was employed. In the remaining parts of this section, the effects of input scaling, distortion function scaling, quantization threshold, and RNS range upon recognition error rate for this simulation are described.

3.7.2 Effect of Input Scaling on Recognition Error Rate

Figure 3.10 is a plot of the recognition error rate versus the input scaling employed for a fixed Itakura-Saito distortion function scaling by 2^{27}. (Of course, as the input scaling changes, the inverse correlation coefficient scaling is also changed in a complementary way to keep the distortion scaling constant.) Input scaling appears to have little effect on error rate until the scale factor reaches 2^{-4}. Test input values in the 12-bit range probably should not be scaled down by more than 2^3.

IA-72.250



RNS: 73, 71, 67, 61, 59.
DISTORTION FUNCTION SCALED BY 2^{27}.

Figure 3.10 RNS IMPLEMENTATION: RECOGNITION ERROR RATE VS. INPUT SCALING

3.7.3 Effect of Distortion Function Scaling on Recognition Error Rate

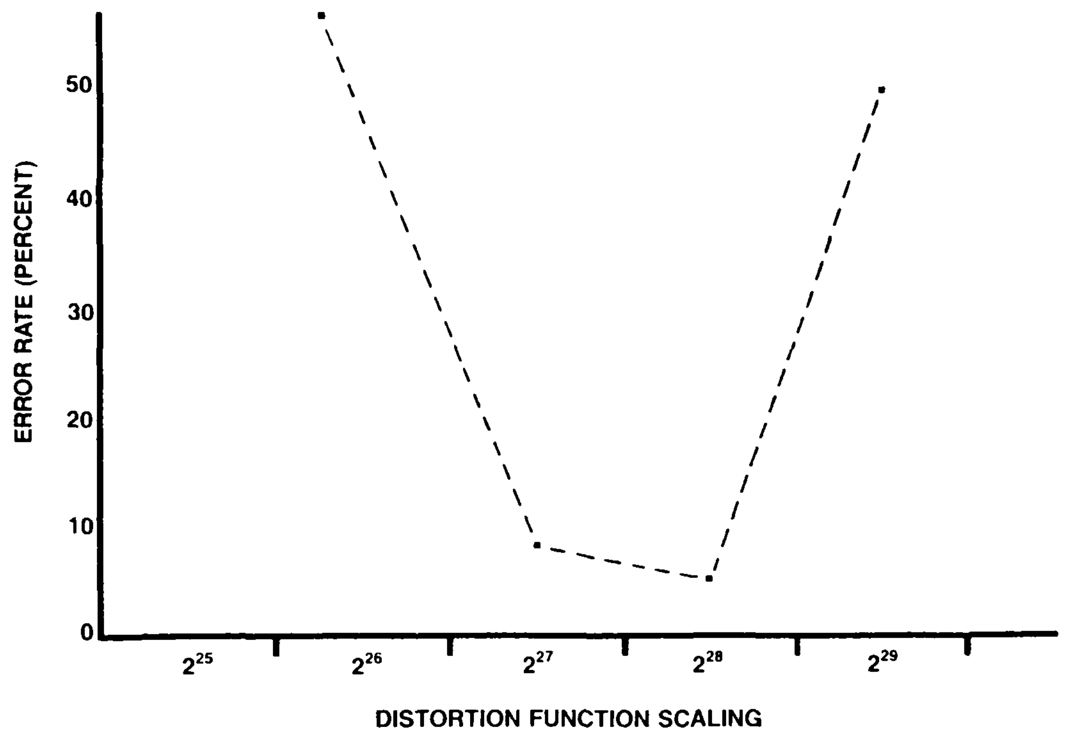
Figure 3.11 is a plot of the recognition error rate versus the Itakura-Saito distortion function scaling employed for a fixed input scaling by 2^{-2}. (Of course, as the distortion function scaling changes, the inverse correlation coefficient scaling is also changed in a complementary way to keep the input scaling constant.) Not shown is the 90 percent error rate obtained for a distortion function scaling by 2^{25}. The high error rates obtained for the cases 2^{25} and 2^{26} reflect insufficient scaling of the reference inverse correlation coefficients; the high error rate obtained for a distortion function scaling by 2^{29} reflects overflow of the RNS by the distortion calculation.

Another view is presented in figure 3.12, which shows the recognition error rates obtained for various combinations of test input scaling, distortion function scaling, and inverse correlation coefficient scaling (any two of which may be set independently). Acceptable performance was realized for certain combinations resulting in a distortion function scaling of 2^{27} or 2^{28}. It is expected that improved performance would result if the RNS range and distortion function scaling were both increased together.

3.7.4 Effect of Quantization Threshold on Recognition Error Rate

Figure 3.13 shows the effect of the choice of the quantization threshold employed for the second quantization, i.e., from the range of the modulus 73 to a single bit, upon recognition error rate. For these simulations the input values were scaled by 2^{-3} and the distortion function was scaled by 2^{28}. A threshold of 16 gave the

IA-72,251

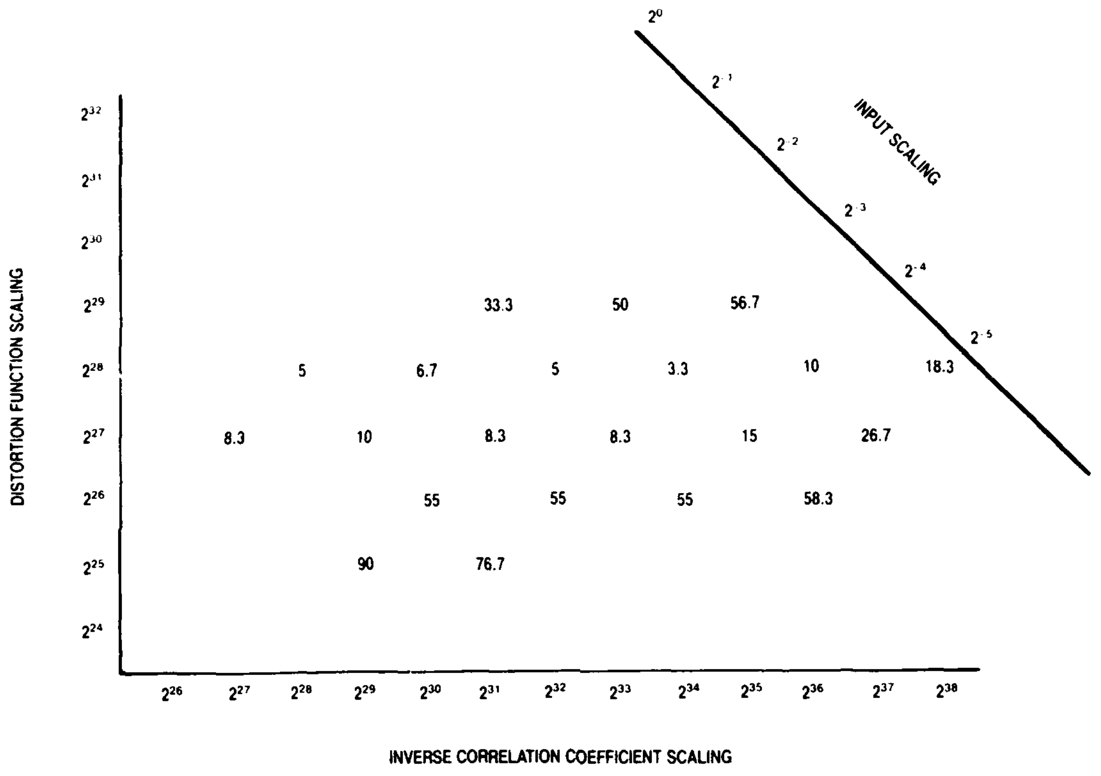


RNS: 73, 71, 67, 61, 59.

INPUT SCALED BY 2^{-2}.

Figure 3.11 RNS IMPLEMENTATION: RECOGNITION ERROR RATE VS. DISTORTION FUNCTION SCALING

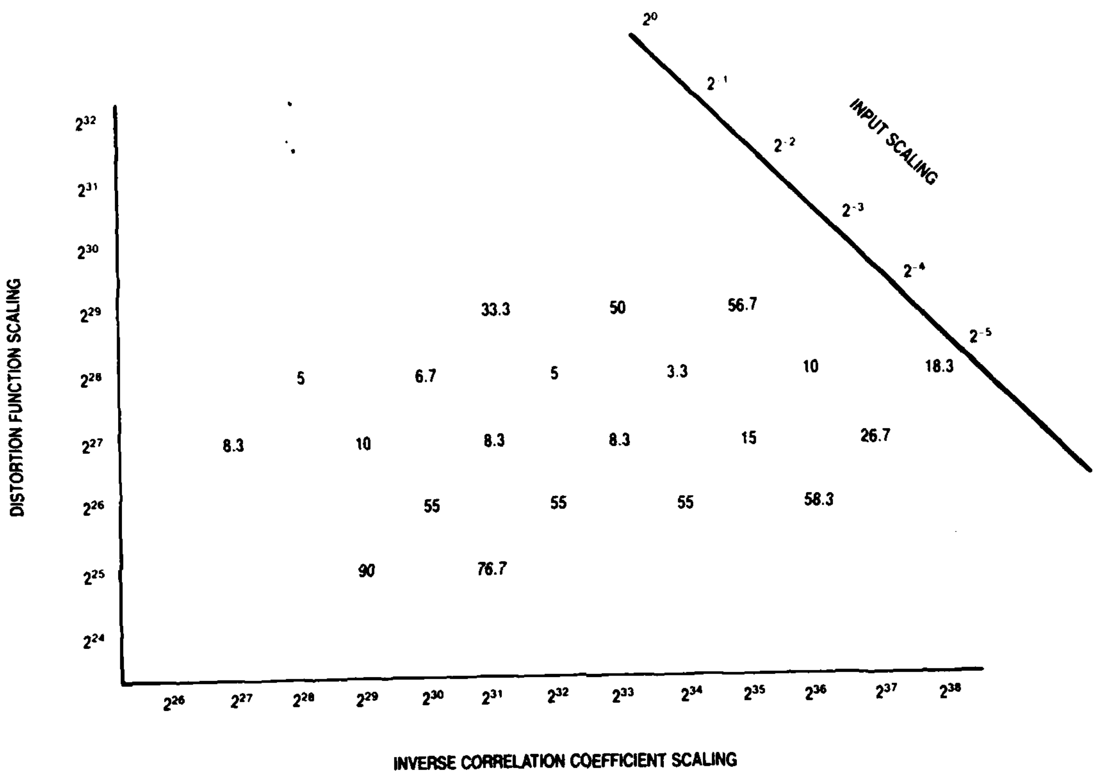
IA-72,534



RNS: 73, 71, 67, 61, 59

Figure 3.12 RECOGNITION ERROR RATES

IA-72,534



RNS: 73, 71, 67, 61, 59

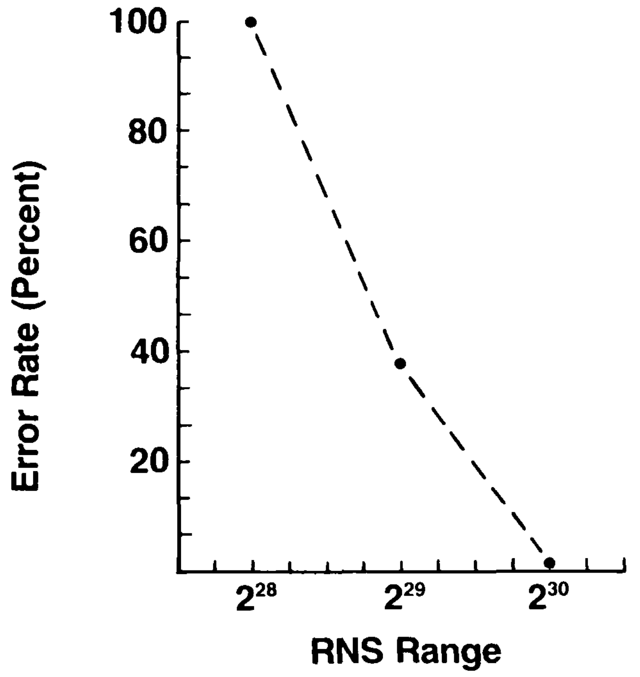
Figure 3.12 RECOGNITION ERROR RATES

best performance, but over a fair range the error rate does not appear to be especially sensitive to this choice. (The results contained in figure 3.12 were all obtained using a threshold of 12.)

3.7.5 Effect of RNS Range on Recognition Error Rate

Figure 3.14 shows the effect of RNS range upon recognition error rate, with a constant scaling of the Itakura-Saito distortion function values by 2^{28}. Test input values were scaled by 2^{-3} and a quantization threshold of 16 was used. Three different residue number systems were employed: (73, 71, 67, 31, 29), with a range of approximately 2^{28}; (73, 71, 67, 31, 59), with a range of approximately 2^{29}; and (73, 71, 67, 61, 59), with a range of approximately 2^{30}. The first RNS gave an error rate of 100 percent, the second an error rate of about 40 percent, and the third an error rate of 1.7 percent. Clearly, the range is important. The RNS range must be sufficient to contain the scaled distortion function values most of the time.

IA-72,252



$$\text{RNS:} \begin{cases} 73, 71, 67, 61, 59 \\ 73, 71, 67, 31, 59 \\ 73, 71, 67, 31, 29 \end{cases}$$

Distortion Function Scaled by 2^{28}
Input Values Scaled by 2^{-3}
Quantization Threshold = 16

Figure 3.14 RNS IMPLEMENTATION: RECOGNITION ERROR RATE VS. RNS RANGE

SECTION 4

SYSTOLIC ARCHITECTURE FOR RNS IMPLEMENTATION

This section discusses the implementation of the autocorrelation sample computation and dynamic time-warping (DTW) algorithm. For the former, a linear systolic array is presented which allows pipelined computation of the autocorrelation coefficients, while accepting a continuous flow of input speech samples, without requiring the insertion of zeros between adjacent frames.

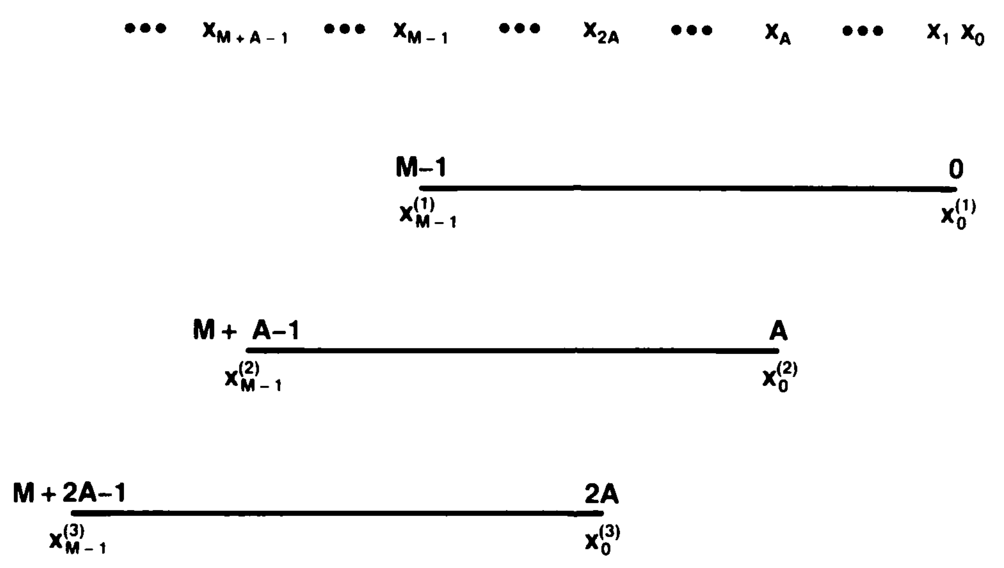
For the DTW algorithm, a two-dimensional pipelined systolic array of processing cells is discussed. Operation is pipelined for both the distortion value and the path metric computation, yielding the score for a pair of test and reference utterances at every step of the operation.

Both arrays are well-suited for RNS implementation, as will be discussed.

4.1 AUTOCORRELATION COMPUTATION

Let $x = \{x_m, m \geq 0\}$ be a set of equally spaced and appropriately windowed samples representing the speech signal. We consider a finite portion of the signal corresponding to a test utterance and partition it into overlapping segments of M samples each as shown in figure 4.1. The ℓ-th segment is denoted $x^{(\ell)} = \{x_m^{(\ell)}, 0 \leq m \leq M-1\}$, and the shift between segments is denoted by A.

IA-71.964



$\underline{r}^{(1)} = (r^{(1)}_p, \ldots, r^{(1)}_0)$

$\underline{r}^{(2)} = (r^{(2)}_p, \ldots, r^{(2)}_0)$

$\underline{r}^{(3)} = (r^{(3)}_p, \ldots, r^{(3)}_0)$

$$r^{(l)}_n = \sum_{m=0}^{M-1} x^{(l)}_m x^{(l)}_{m+n}$$

Figure 4.1. INPUT SEGMENTATION

From each segment a vector $\underline{r}^{(\ell)} = \left(r_P^{(\ell)}, \ldots, r_0^{(\ell)}\right)$ of P+1 autocorrelation coefficients given by

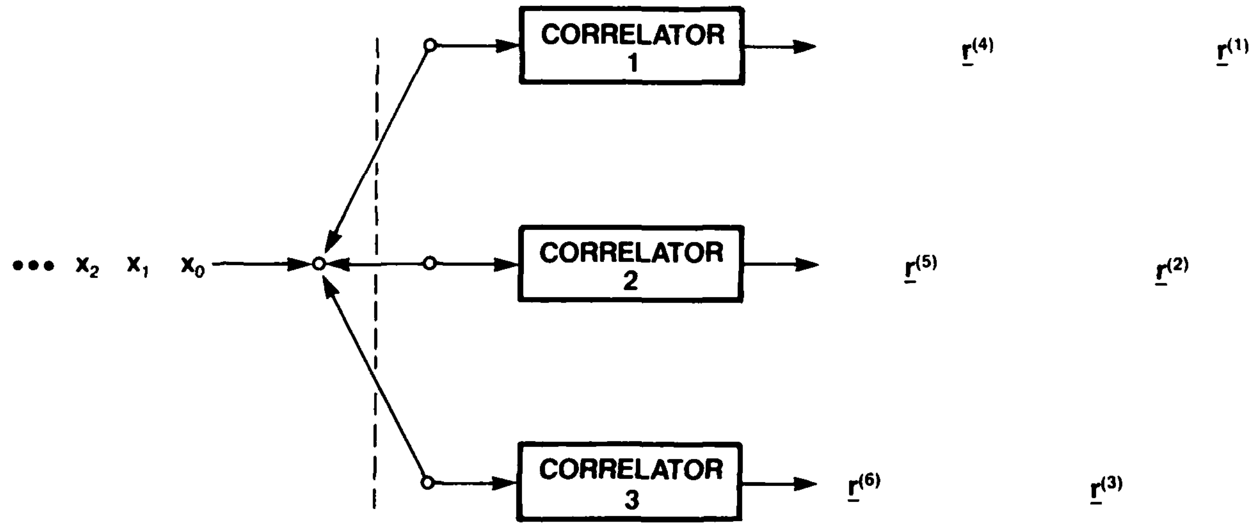
$$r_n^{(\ell)} = \sum_{m=0}^{M-1-n} x_m^{(\ell)} x_{m+n}^{(\ell)} \qquad (4.1)$$

is to be extracted.

The values of M and A that we implement are 180 and 80 samples respectively. With these parameters, at most three segments overlap at any time, and hence three correlators will be needed if the correlation vectors are to be computed in a pipeline.

Figure 4.2 shows three correlators which switch the input data stream on and off precisely at the beginning and end of their respective segments. Thus, $\underline{r}^{(1)}$ is computed by correlator 1, $\underline{r}^{(2)}$ by correlator 2 and $\underline{r}^{(3)}$ by correlator 3. When the first sample of segment 4, $x_0^{(4)} = x_{3A}$, appears, all samples of segment 1 have entered correlator 1, so it is ready to receive segment 4. Care has to be taken so that in correlator 1 samples from segment 4 do not mix with those of segment 1. This can be accomplished by the linear systolic architecture which we now describe.

IA-71.960



••• x_2 x_1 x_0
CORRELATOR 1
CORRELATOR 2
CORRELATOR 3
$\underline{r}^{(4)}$ $\underline{r}^{(1)}$
$\underline{r}^{(5)}$ $\underline{r}^{(2)}$
$\underline{r}^{(6)}$ $\underline{r}^{(3)}$

Figure 4.2. AUTOCORRELATION COMPUTER

4.2 LINEAR SYSTOLIC ARRAY FOR AUTOCORRELATION COMPUTATION

Figure 4.3 shows a linear systolic array composed of P + 1 processing cells and P + 1 delay elements forming the output register. The input samples enter into the leftmost and rightmost cells and are passed along to the next cell to the right and to the left, respectively. At each step every cell forms the product of its two inputs and adds it to its contents. After all 180 samples have been operated on, the autocorrelation coefficients will be in the cell registers and at that point they can be passed down to the output register and circulated out to the right.

A detailed example with 5 cells (P = 4) is developed at successive times in figure 4.4. Input samples are interleaved with zeros, and the left input enters the array first, from the left, so that it meets the first sample from the right input at cell 0. The figure shows the state of the array at consecutive time instants. As the signals progress through the array, sums of products accumulate in each cell, with the computation of $r_n^{(\ell)}$ taking place in cell n, $0 \leq n \leq P$.

Figure 4.5 shows the last few samples of a segment, followed by zeros. At the end of the process, the n-th autocorrelation coefficient resides in cell n, $0 \leq n \leq P$, at which point it can be fed out serially in the manner already described, or in parallel (technically violating the systolicity of the operation), if needed.

IA-71,961

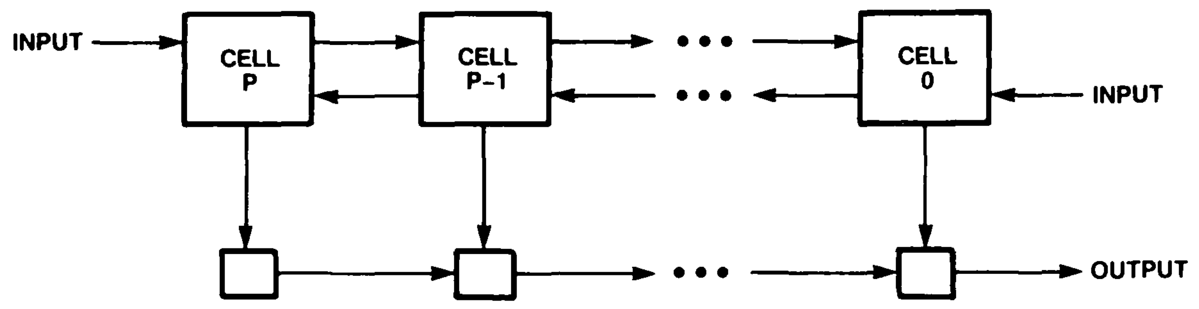


Figure 4.3. SYSTOLIC AUTOCORRELATION COMPUTER

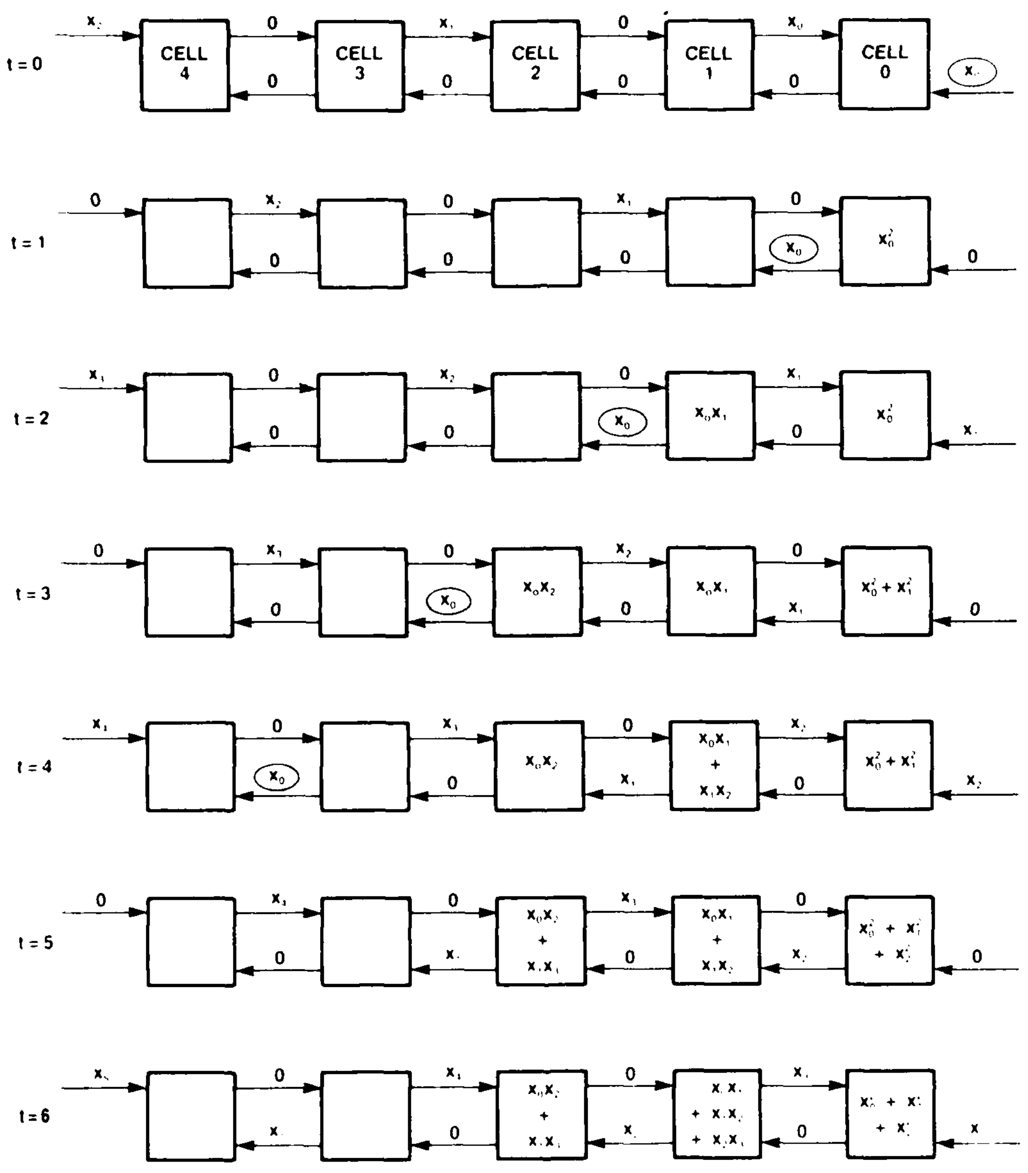
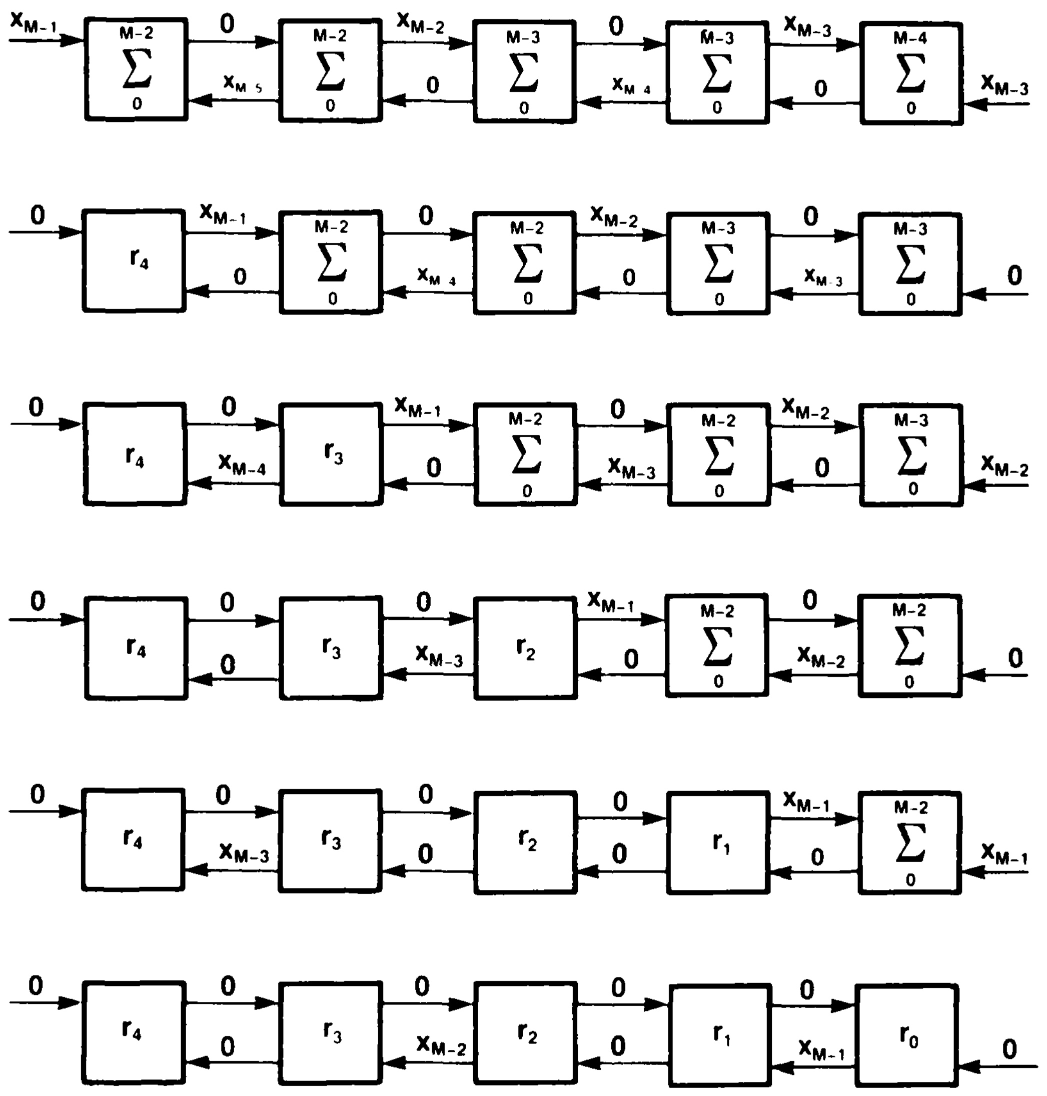


Figure 4.4. AUTOCORRELATION COMPUTATION



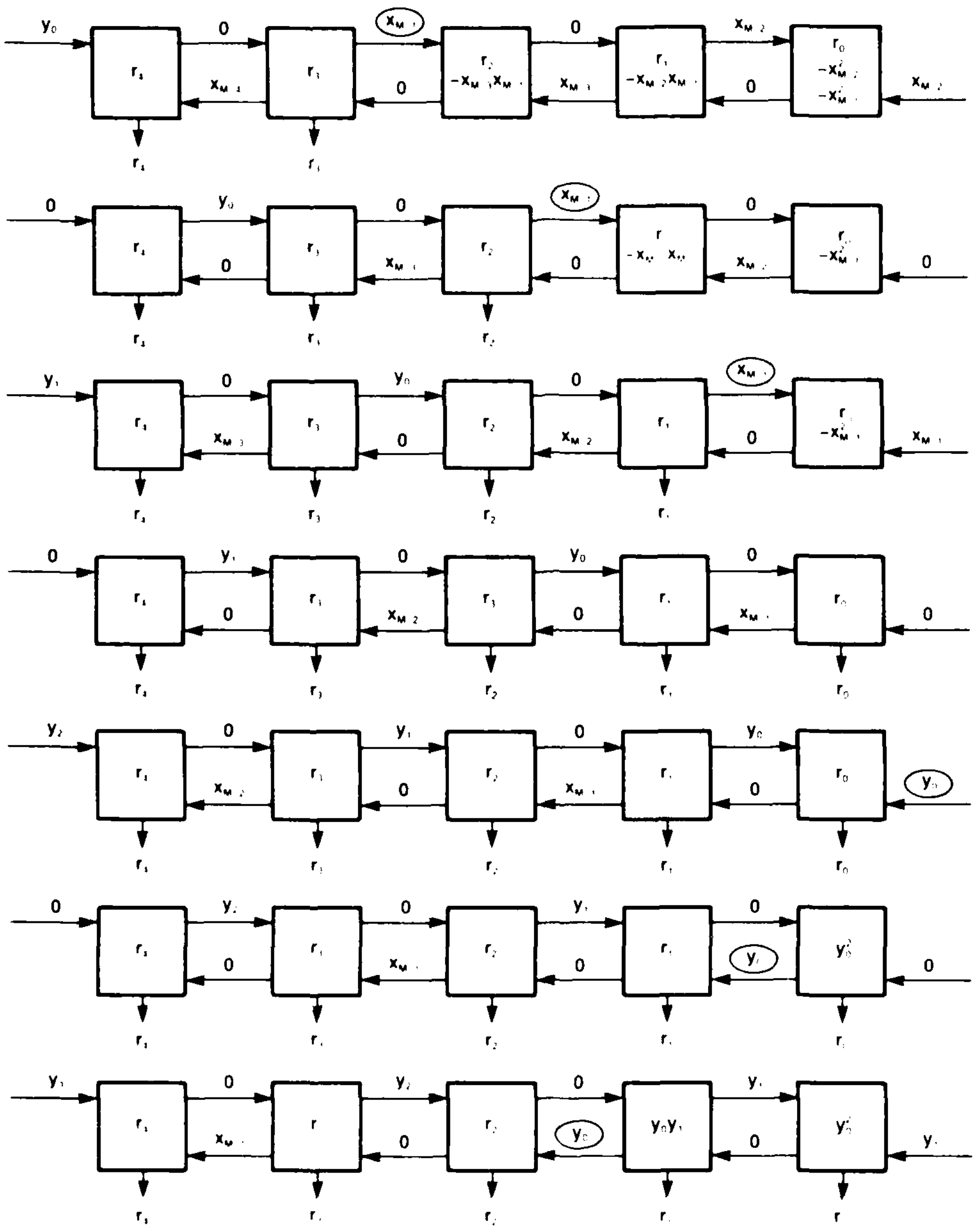
IA-71,968

Figure 4.5 AUTOCORRELATION COMPUTATION: FINAL STEPS

In pipelined operation, samples of a new segment may (depending on the values of M and A) enter the array before the computation of the present segment's autocorrelation vector has been completed, and a cell may receive inputs from both segments at the same time, in which case they should be ignored. This can be taken care of by attaching a control bit to the first sample of the left input segment and to the first sample of the right input segment. The left control bit appears at any given cell when the last term in the sum is being computed and it instructs the cell to ignore subsequent inputs, both from the left and from the right, until the control bit in the first sample of the new segment coming from the right appears, instructing the cell to output its contents, clear its accumulator and resume operations, as the computation of the corresponding coefficient of the new segment begins.

Figure 4.6 illustrates the interplay of the two control bits. The old segment is labeled x_0, x_1, ..., x_{M-1}, and the new segment y_0, y_1, ..., y_{M-1}. The samples carrying control bits are circled. The arrows emerging from the bottom of the cells indicate that the result has become available.

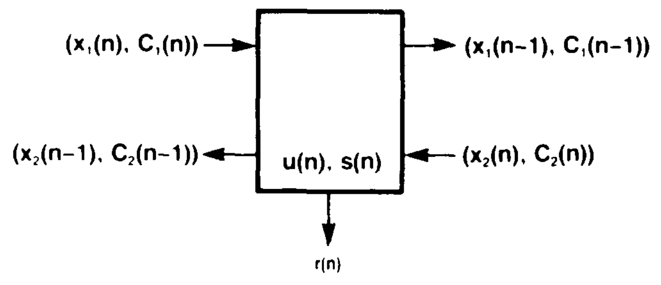
The equations governing the operation of each cell are shown in figure 4.7. The left input is indicated by a subscript 1 and the right input by a subscript 2; the time index is n and is shown in parentheses. The corresponding control bits are $C_1(n)$ and $C_2(n)$. The quantities u(n) and s(n) are state variables. The variable u(n) is used in the operation of the control bits; it is initially a zero and it changes its value every time a control bit appears. The result is that u(n) = 1 precisely when the cell is computing, and u(n) = 0 between the computations of consecutive segments, when the cell is ignoring its inputs. The variable s(n) stores the partial sums and r(n) is the output.



IA-71,970

Figure 4.6. AUTOCORRELATION COMPUTATION: TWO CONSECUTIVE INPUT SEGMENTS

IA-71.959



$$u(n+1) = \begin{cases} 1 \text{ IF } n+1 < 0 \\ u_n \text{ IF } C_1(n) = 0 \\ u_n' \text{ IF } C_1(n) = 1 \text{ OR } C_2(n) = 1 \end{cases}$$

$$s(n+1) = \begin{cases} 0 & \text{IF } n+1 < 0 \\ s(n) + x_1(n)\, x_2(n) & \text{IF } u(n) = C_2(n) = 0 \\ s(n) & \text{IF } u(n) = 1,\ C_2(n) = 0 \\ x_1(n)\, x_2(n) & \text{IF } C_2(n) = 1 \end{cases}$$

$$r(n+1) = \begin{cases} s(n+1) \text{ IF } C_2(n) = 1 \\ r(n) \text{ IF } C_2(n) = 0 \end{cases}$$

Figure 4.7. PROCESSING ELEMENT IN AUTOCORRELATION ARRAY

4.2.1 RNS Hardware Concept

Appendix A discusses the fundamentals of residue number systems.

Consider implementing equation (4.1) in an RNS with prime moduli p_1, p_2, ..., p_ℓ, and range M. First, the input must be converted to residue form. Then, for each k, equation (4.1) is computed modulo p_k. This requires one set of correlators like the one in figure 4.2 for each modulus. The output, the residues of the autocorrelation coefficients, are then used for the computation of the local distortion.

The configuration of the correlator using ℓ residue channels is shown in figure 4.8. For each channel, a copy of the control bits c_1 and c_2 must accompany the inputs.

Figure 4.9 shows the structure of each correlator cell. Each cell uses one mod p multiplier and one mod p adder, six data registers, a few gates and switches and some control logic which controls the output flow.

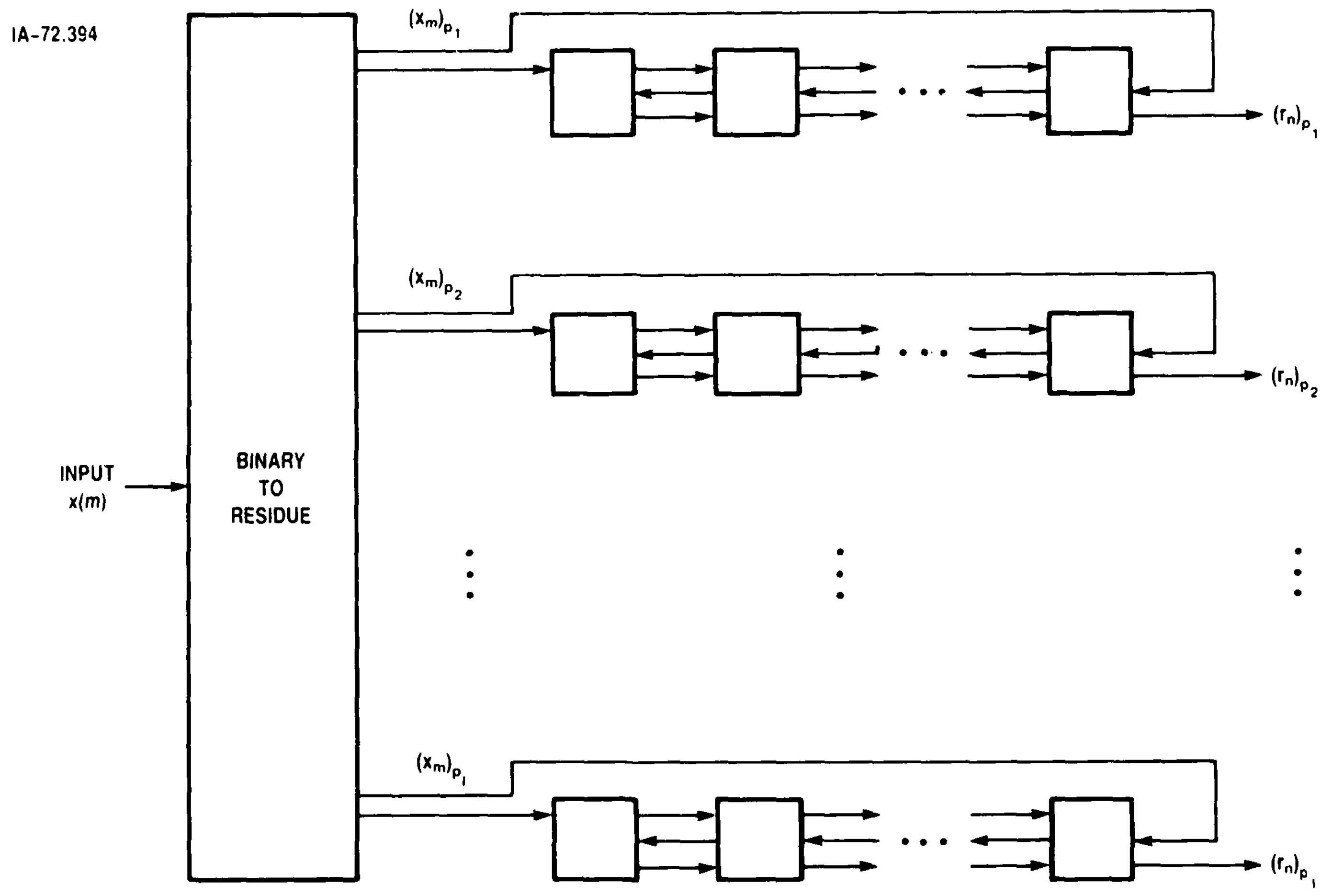


Figure 4.8. SYSTOLIC RNS AUTOCORRELATION COMPUTER

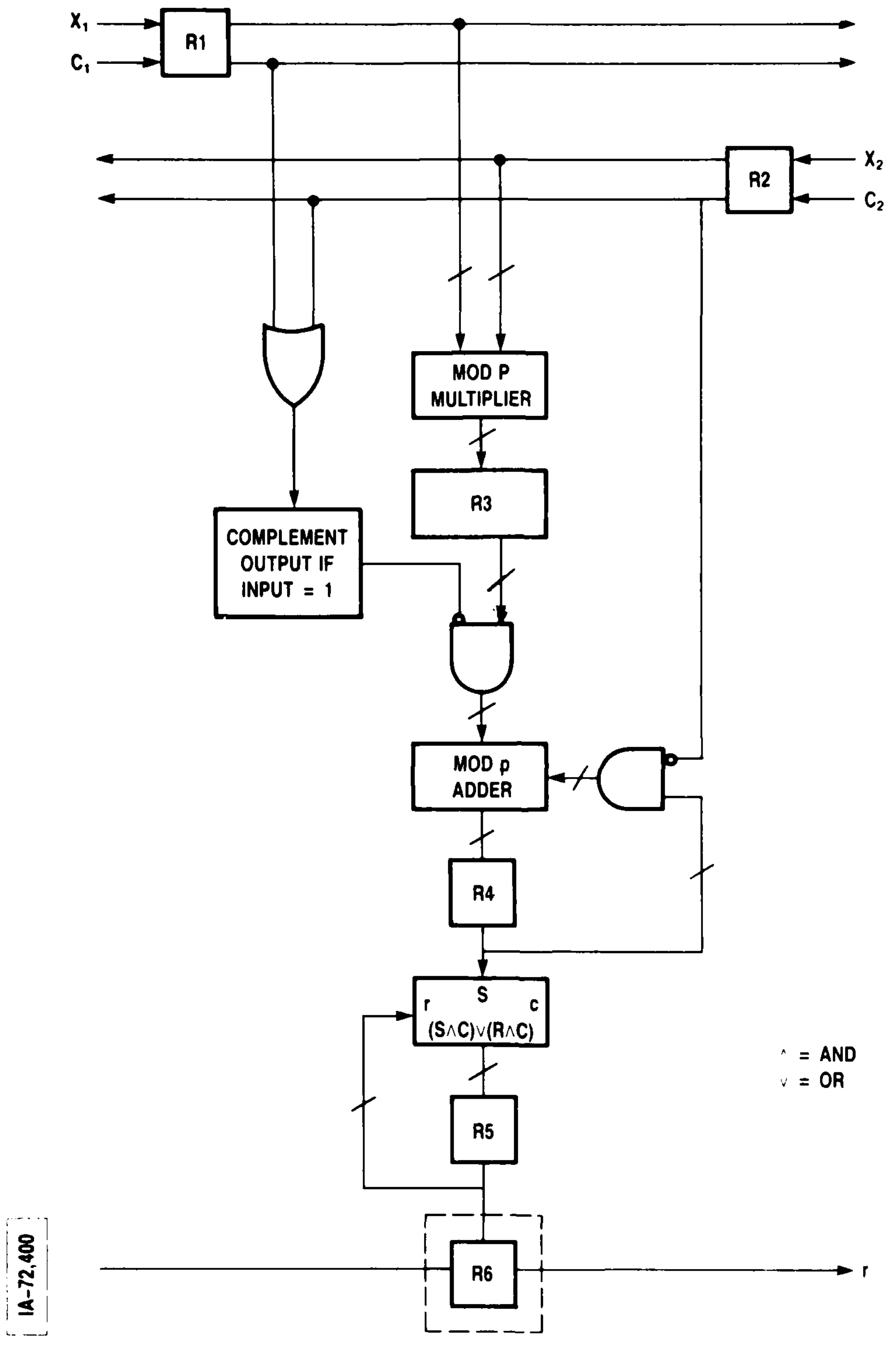


Figure 4.9. SYSTOLIC CORRELATOR CELL MOD p

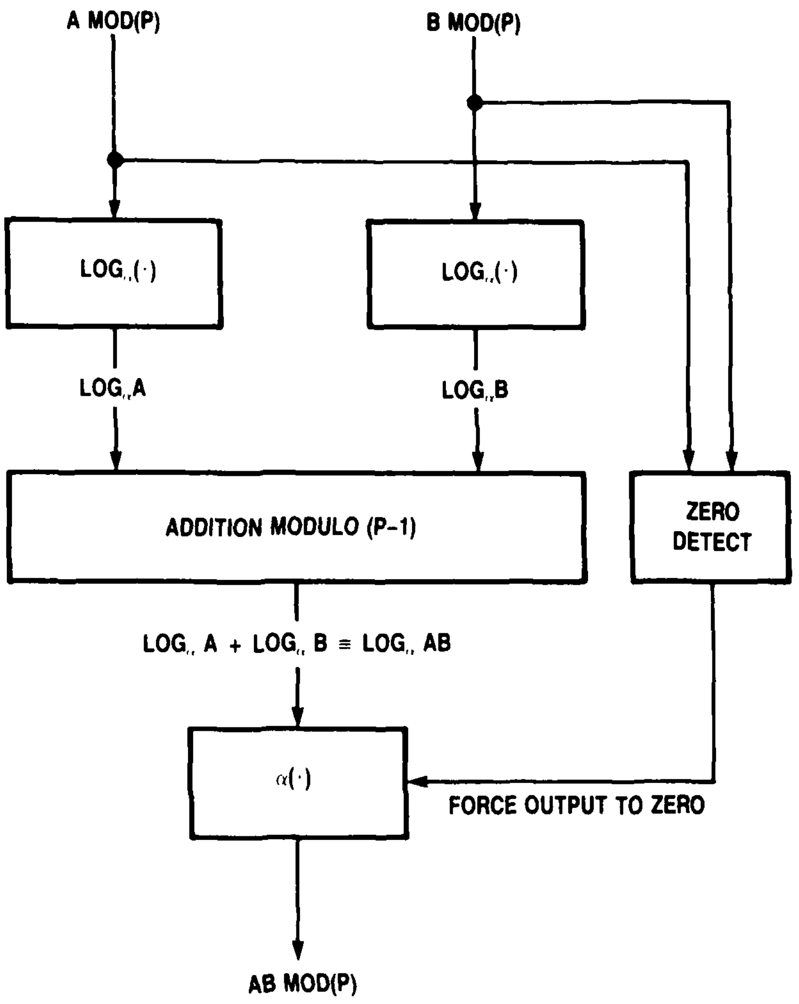
The detailed designs of the modulo p adders and multipliers, as well as the binary-to-residue converter, were carried out by MITRE's Integrated Electronics project to implement a transversal equalizer. They developed a logarithmic mod p (p is prime) multiplier. To multiply two residues a and b modulo p, their logarithms base α (where α is another fixed element of the field) are computed. The two quantities are added modulo p and the inverse logarithm of the result is then computed. Symbolically,

$$\log_\alpha AB = \log_\alpha A + \log_\alpha B.$$

Figure 4.10 shows a block diagram of the multiplier that was developed under the Integrated Electronics project.

A great savings is realized if the logarithms of the input signals are computed before they enter the array, for then only the inverse logarithm needs to be computed in each cell. This idea was used by MITRE's Integrated Electronics project to implement a transversal filter.

To select the moduli we assume a frame length of 180 samples, an LPC model of 13 poles and no zeros, and input speech samples in the range $[-2^8, 2^8]$. An upper bound on the absolute value of the size of the autocorrelation coefficients is then given by $180(2^8)^2$, or a range of about $[-2^{24}, 2^{24}]$. The distortion values will then lie in the range $[-2^{56}, 2^{56}]$, which is required to contain the dot product of two feature vectors. However, it was determined by simulation that, for virtually all inputs occurring in practice, the required dynamic range is only $[-2^{33}, 2^{33}]$, which is spanned by the five seven-bit moduli 103, 107, 109, 113, and 127.



1A-72.397

Figure 4.10. MODULO P MULTIPLIER: LOGARITHMIC IMPLEMENTATION

4.3 SYSTOLIC ARRAY FOR DTW COMPUTATIONS

Once the correlation vectors for the test segments have been computed (and the logarithm of the process gain, if the Itakura-Saito distortion function is used), the DTW computations can be carried out in a pipelined two-dimensional processing array. It is assumed that the inverse-autocorrelation coefficients (and logarithm of the process gain) for the reference segments are available in a stored reference library and need no computation during the speech recognition processing.

The DTW computations, as noted in section 3, separate into computation of the distortion d_{ij} between the ith test segment and jth reference segment and the actual path metric computations associated with the dynamic programming algorithm to find the metric of the shortest path.

The local distortion computation (for the Itakura-Saito metric) involves formation of a scalar product between the test and reference vectors of correlation and inverse correlation values with the addition of a constant term dependent on the logarithmic ratio of the process gains, followed by quantization within RNS as discussed in section 3. The dynamic programming algorithm uses the quantized distortion values in a decision-directed algorithm that preserves only the best path metrics at each iterative step. As long as the distortion values that are needed in the path metric computations are computed in advance of their need, both the distortion and path metric computations can be carried out in the same systolic array. Since the distortion values are either absorbed into the path metric computations or discarded as they are used, the computation can progress through the array as a wavefront leaving empty cells in its wake to provide a fully pipelined capability.

Below we describe the flow of these computations in a two-dimensional array of only a few cells for purposes of illustration. The architecture described is readily extrapolated to a larger array.

4.3.1 Computation of the Array of Distortion Values

Once the test utterance has been partitioned into n overlapping segments, the jth segment being represented by a vector $\underline{r}(j)$, $1 \leq j \leq n$ of P + 1 autocorrelation coefficients of the test segment, and the reference utterance has been partitioned into m segments represented by vectors $\underline{u}(i)$, $1 \leq i \leq m$, of P + 1 inverse-autocorrelation coefficients of reference segments, an $m \times n$ grid is formed with the distortion (cost) d_{ij} at each grid point (i,j) being the scalar product of $\underline{r}(j)$ and $\underline{u}(i)$ plus a constant term dependent on the logarithm of the ratio of the process gains for the Itakura-Saito distortion function. The DTW algorithm computes the least cost among paths between grid points (1,1) and (m,n), the cost being the sum of all distortions d_{ij} encountered along the path. For the Itakura-Saito distortion, we will concentrate first on the pipelined calculation of the scalar product of the autocorrelation vectors and will show later how to incorporate the constant term in the architecture.

After computation of a distortion value, it must be quantized to a lower range. The details of such a quantization were discussed in section 3.4.3.

In order to pipeline the distortion computation, the correlation vectors $\underline{r}^{(\ell)}(j)$ and $\underline{u}^{(\ell)}(i)$ flow into a rectangular array of computational cells as shown illustratively in figures 4.11 and 4.12(a) through 4.12(g), where a time sequence of data flow is presented for a square array of nine cells. Figure 4.13 shows the arrangement of the various distortion functions contained in the array at a given instant of time; the superscripts in both cases refer to the collection of a set of segments associated with a particular utterance.

Since the path computations for each pair of test and reference utterances will proceed as a wavefront making computations on successive diagonals, the deletion of previously used distortion values allows pipelining to the extent that distortion functions associated with a number of utterances equal to the number of diagonals can be present at any given time, with the path computations being pipelined along successive diagonals.

In figures 4.11 through 4.13, vectors on the same diagonal carry the same superscript and correspond to the same test or reference utterance. The data proceed in straight lines (test vectors horizontally and reference vectors vertically) through the array, and at each time instant each cell (i,j) computes

$$\underline{r}^{(\ell)}(j) \cdot \underline{u}^{(\ell)}(i) = \sum_{n=1}^{P} r_n^{(\ell)}(j)u_n^{(\ell)}(i)$$

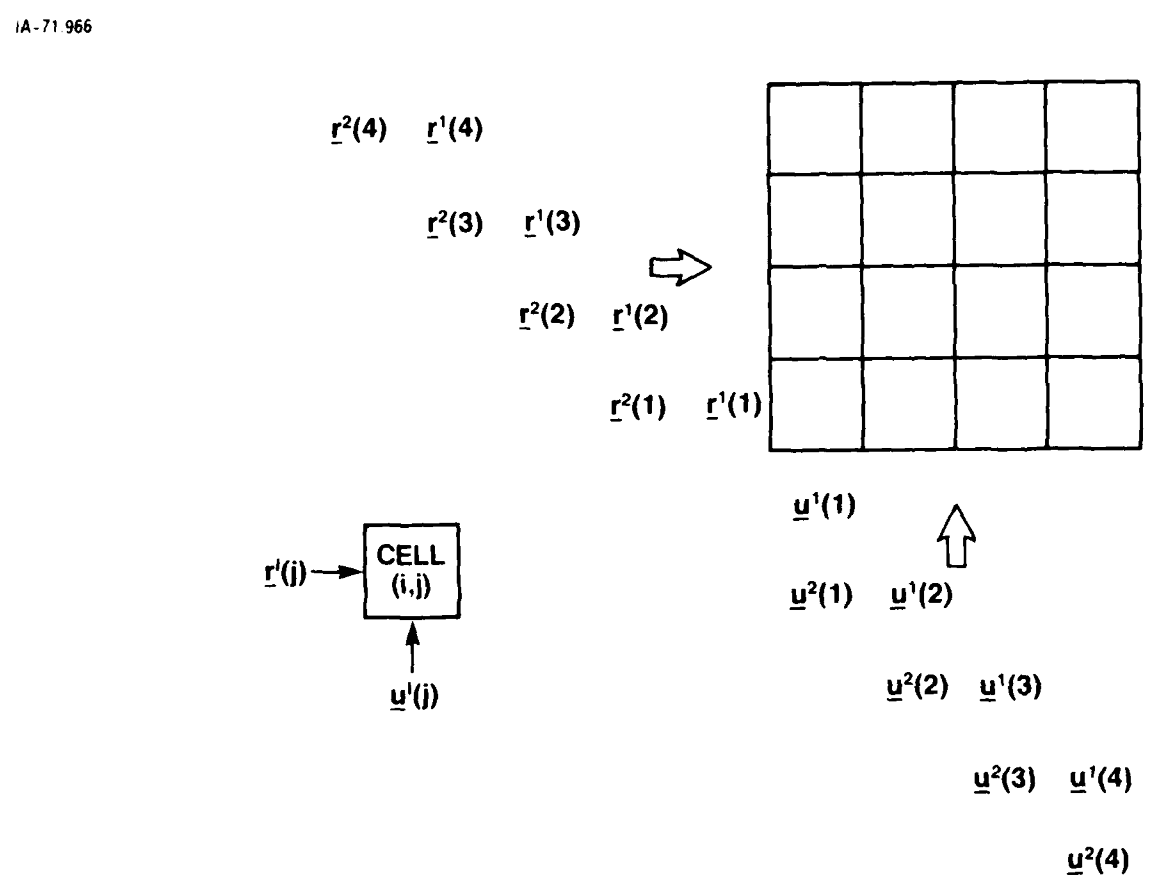

IA-71 966
$\underline{r}^2(4)$ $\underline{r}^1(4)$
$\underline{r}^2(3)$ $\underline{r}^1(3)$
$\underline{r}^2(2)$ $\underline{r}^1(2)$
$\underline{r}^2(1)$ $\underline{r}^1(1)$
$\underline{u}^1(1)$
$\underline{u}^2(1)$ $\underline{u}^1(2)$
$\underline{u}^2(2)$ $\underline{u}^1(3)$
$\underline{u}^2(3)$ $\underline{u}^1(4)$
$\underline{u}^2(4)$
$\underline{r}^l(j)$
CELL (i,j)
$\underline{u}^l(j)$

Figure 4.11. DTW INPUT DATA FLOW

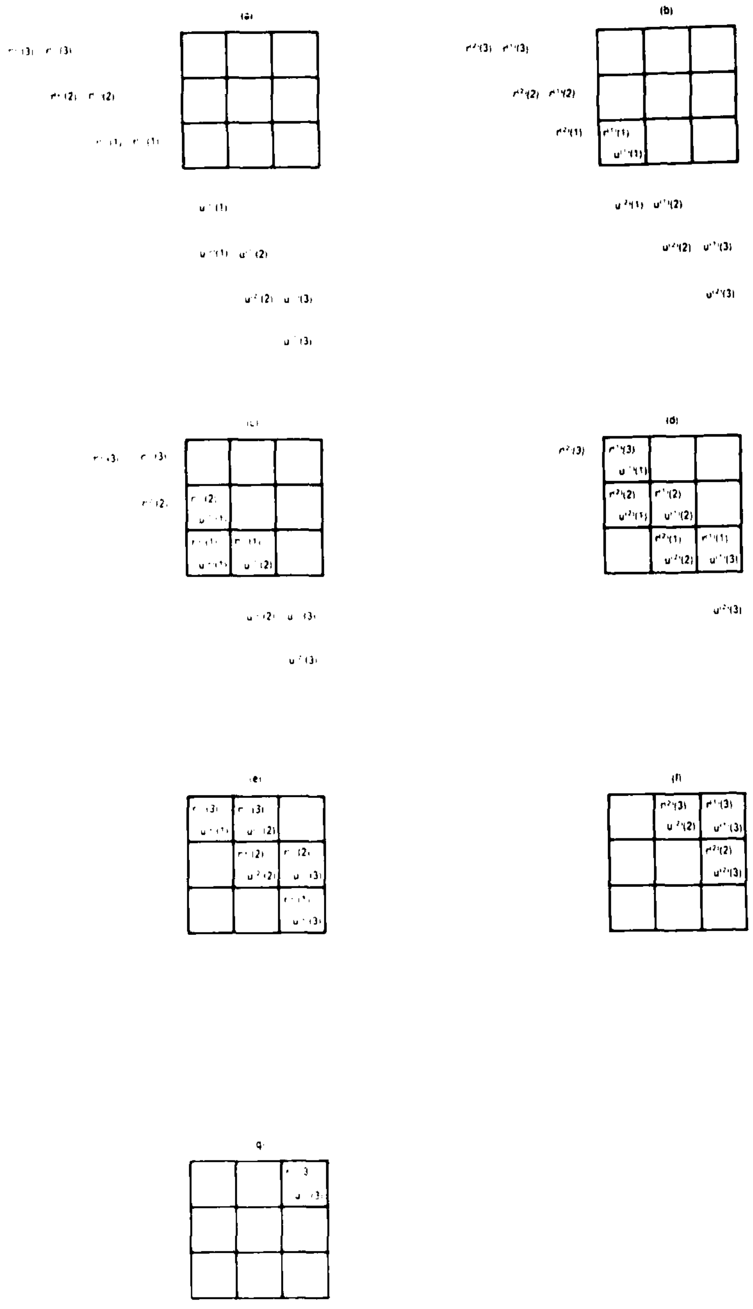


Figure 4.12 SYSTOLIC COMPUTATION OF LOCAL DISTORTION

IA-71.967

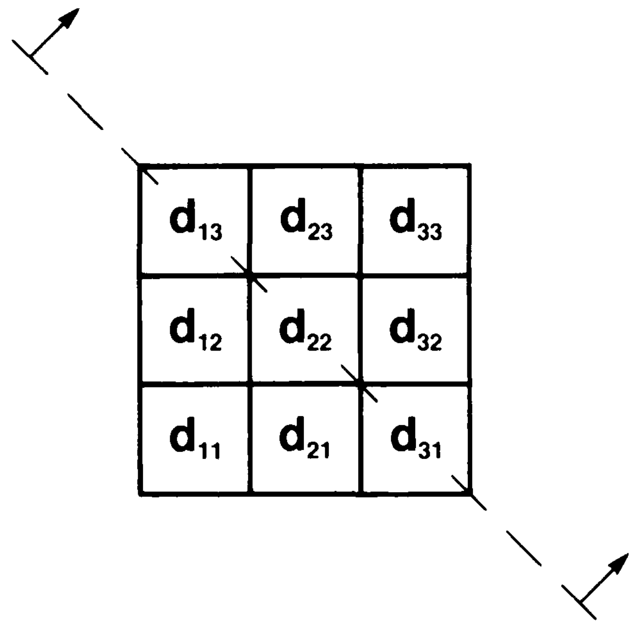


Figure 4.13. COMPUTATIONAL WAVEFRONT

the scalar product of $\underline{r}^{(\ell)}(j)$ and $\underline{u}^{(\ell)}(i)$, the vectors appearing at its inputs, to form the principal term in the Itakura-Saito distortion. In this manner, all cells on any given diagonal perform computations on the same pair of test and reference utterances.

Performing the distortion computation in RNS suggests a separate grid for each modulus. The inputs to the kth array are the residues modulo p_k of the feature vectors, $\underline{u}(i)$, $\underline{r}(j)$, from which the residues modulo p_k of the local distortions d_{ij} are computed. In this operation the different arrays work independently and in parallel. The need for quantization, however, will require communication of the residue values prior to path computations.

Figure 4.14 is a simplified block diagram of the computation of the local distortion in a residue channel modulo p. As the components of the test and reference correlation vectors enter the multiplier, the modulo p product of corresponding components accumulates. A flag bit can be attached to $u_0(i)$ to clear the accumulator at the beginning of every new segment.

With our 13-pole model, the computation of

$$\underline{u}(i) \cdot \underline{r}(j) \equiv \sum_{n=0}^{12} u_n(i)\, r_n(j) \bmod p \qquad (4.2)$$

requires 13 multiplications and 12 additions, which, if performed in a pipeline, require 13 τ_m + $12\tau_s$ seconds, where τ_m and τ_s are the times, in seconds, required for one residue multiplication and one residue addition, respectively.

IA-72.395

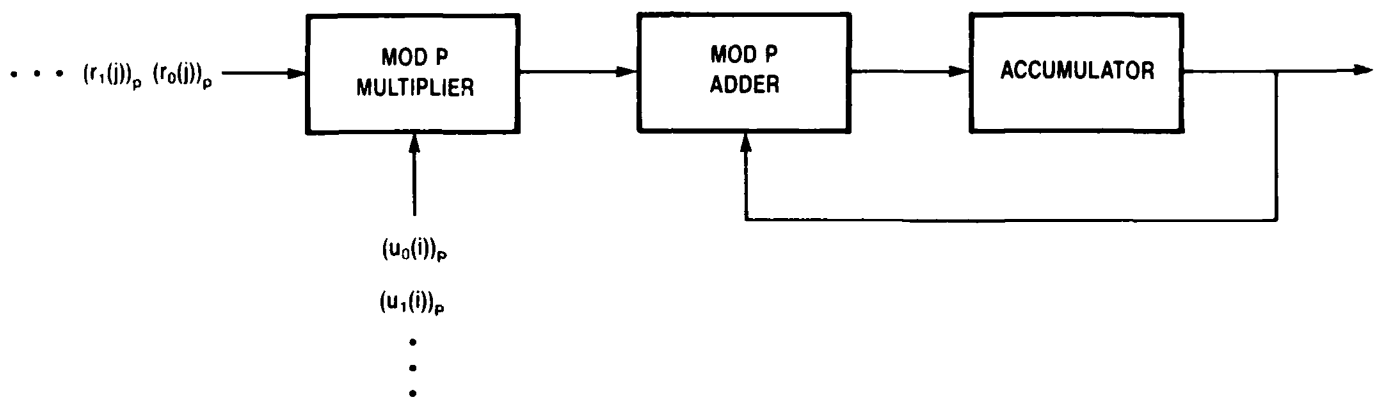


Figure 4.14. LOCAL DISTORTION COMPUTATION

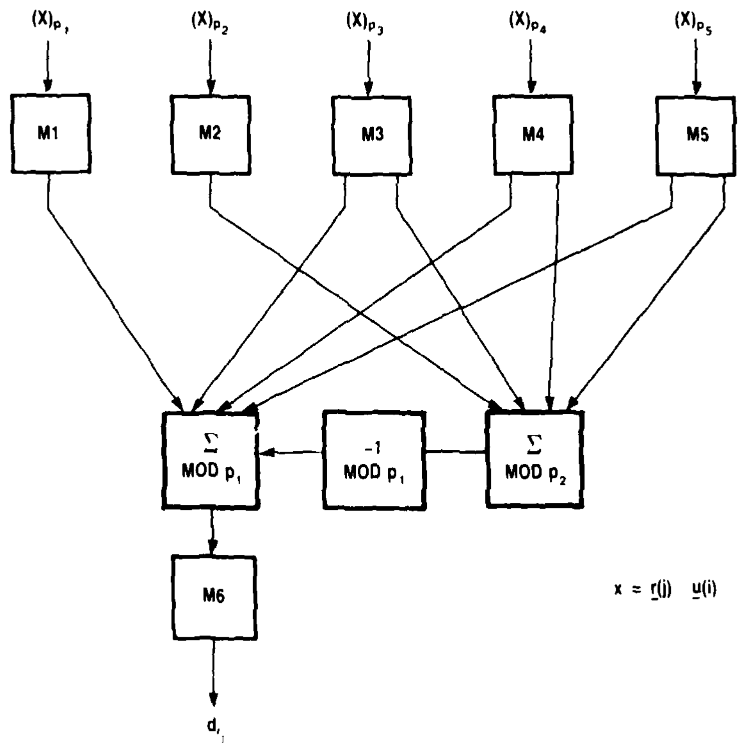
4.3.1.1 Quantization of the Distortion Values

Next, the scalar product of $\underline{u}(i)$ and $\underline{r}(j)$ is quantized to a smaller range spanned by three moduli. Quantization operates on all residues of the scalar product. (A block diagram of this operation is given in figure 4.15). At this point, the constant terms $\log\sigma_f^2$ and $\log\sigma_g^2$ which may be attached to $\underline{r}(j)$ and $\underline{u}(i)$ respectively before entering the array, may be added, converted to the three-modulus RNS and added to the quantized version of $\underline{u}(i) \cdot \underline{r}(j)$ to complete the computation of d_{ij}.

The computation of $-\log\sigma_f^2$ can be performed outside RNS in parallel with the computation of $\underline{r}(j)$. The term $\log\sigma_g^2$ can be stored with $\underline{u}(i)$ in the reference library. A method of logarithm generation is given in [10].

Quantization involves all the residues of $\underline{u}(i) \cdot \underline{r}(j)$, so communication between residue channels is required. Each residue of $\underline{u}(i) \cdot \underline{r}(j)$ is entered into a table, so in our five-modulus RNS there are five tables. The outputs of the first two tables are called Q_1 and Q_2, and lie in $[0,p_1-1]$ and $[0,p_2-1]$, respectively. Each of the other three tables outputs two symbols, a_k, b_k, $k = 1, 2, 3$, where a_k is in $[0,p_1-1]$ and b_k is in $[0,p_2-1]$. Then the quantities

$$s_1 = Q_1 + \sum_{k=1}^{3} a_k \bmod p_1 \qquad (4.3)$$



IA-71,956

Figure 4.15. RNS QUANTIZER

and

$$s_2 = Q_2 + \sum_{k=1}^{3} b_k \bmod p_2 \tag{4.4}$$

are computed. After that, s_2 is reduced modulo p_1 and

$$t = |s_1 - s_2|_{p_1} \tag{4.5}$$

is computed. This quantity is then entered into a table whose output is the quantized product.

For one quantization, five table lookups are first performed, in parallel, to produce eight quantities. These are separated into two groups of four quantities, and all four quantities in each group are added by a tree adder. In this manner s_1 and s_2 are produced. Next $|-s_2|_{p_1}$ is computed from s_2 by a table lookup. Finally, an addition is performed to compute t and one more table lookup is done to give the final result. The time required to perform one quantization is then $3\tau_t + 3\tau_s$ seconds, where τ_t is the time required for one table lookup and, as before, τ_s is the time required for one residue addition.

4.3.2 Shortest Path Computations

Once the distortion values are available for use, a dynamic programming algorithm is used to find the shortest, or least distortion, path through the DTW grid. Since the distortion values are quantized to a few levels, as discussed in section 3, the path computations can be carried out in RNS with path differences generally small enough to be contained within the largest modulus of the RNS, occasional overflows not causing catastrophic harm.

To discuss a systolic architecture for these computations it will be convenient to temporarily set aside the pipelining of the distortion computations discussed in section 4.3.2. For the present discussion, we will assume that all the distortion values for a particular pair of utterances are available in the array and describe the pipelined data flow of the path computations in the same array. It will then be evident that both the distortion and path computations can be synchronously pipelined in the same DTW array.

The optimality principle of dynamic programming states that if a path of minimal cost between two points a and c passes through point b, then the portion of the path that goes from point a to point b is optimal too, among paths from a to b. In accordance with this principle, in the DTW algorithm, cell (i,j) computes the cost of the optimal path starting at (1,1) and ending in itself. This is done iteratively, all cells along the same diagonal computing at the same time. Cell (1,1) computes d_{11} and calls it c_{11}, the minimum cost to get to cell (1,1). Next the cells on the second diagonal, cells (2,1) and (1,2), compute d_{21} and d_{12} respectively and add it to c_{11} to produce c_{21} and c_{12}. Then the cells on the third diagonal do the same thing. This time cell (2,2) considers two possible cells from which a path can emanate, namely cells (2,1) and (1,2) and computes $c_{22} = d_{22} + \min(c_{12}, c_{21})$. Global constraints will restrict the number of cells to be considered, while local constraints will restrict the number of paths to each cell.

In general, according to type 3 local path constraints discussed in section 3, each cell considers at most four cells from which a path to it can emanate, i.e., cell (i,j) computes

$$c_{ij} = d_{ij} + \min\left(c_{i-2,j-1} + d_{i-1,j},\ c_{i-2,j-2} + d_{i-1,j},\ c_{i-1,j-1},\ c_{i-1,j-2}\right). \qquad (4.6)$$

For this to be possible, the four path costs must be available at the input of cell (i,j) when the cells on its diagonal are ready to compute. This is clearly possible since the four path costs are on diagonals that have already completed computation. In the physical array all data move horizontally or vertically, and diagonal communication--e.g., the communication of $c_{i-1,j-1}$ from cell (i-1,j-1) to cell (i,j)--is done through cell (i,j-1). At each step on such a path, the data advance one diagonal. All data being operated on, or computed, corresponding to one pair of utterances, lie on the same diagonal.

From this observation, it follows that the distortion values on a given diagonal need not be available until the computational wavefront has reached that diagonal and this can be managed by the pipelined scheme already discussed for the distortion computations. Hence, the DTW can be completely pipelined, with each diagonal handling one pair of utterances. From cell (m,n) the scores of the pairs of utterances compared will then emerge one-by-one.

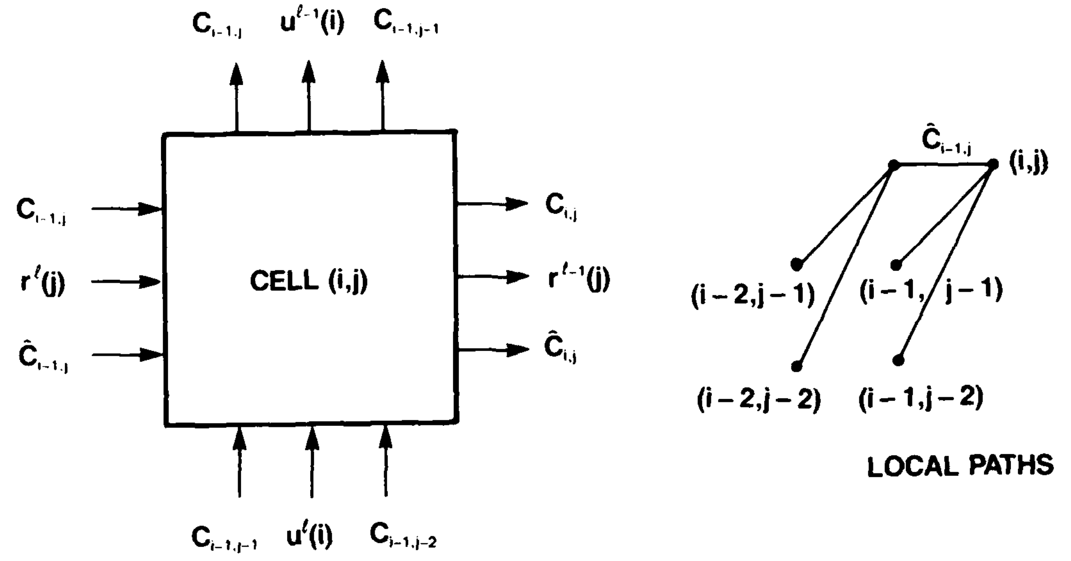
Figure 4.16 shows a typical cell in the DTW grid. The computation of c_{ij} is done in two steps so that actually only three quantities previously computed are fed to each cell ($\hat{c}_{i-1,j}$ is the minimum of $c_{i-2,j-1} + d_{i-1,j}$ and $c_{i-2,j-2} + d_{i-1,j}$). Also, $c_{i-1,j}$ is not used by cell (i,j) but is passed along from cell (i-1,j) to cell (i,j+1). Similarly, $c_{i-1,j-1}$ is passed to the cell above, after it has been used by cell (i,j). Finally, $\hat{c}_{ij}$ is computed and passed to cell (i+1,j). Thus, in addition to inputs $\underline{r}(j)$ and $\underline{u}(j)$, which get passed along after their scalar product is computed, each cell accepts four inputs and produces four outputs.

Figure 4.17 shows a sequence of data flow for the path computation between two utterances in a 3-by-3 DTW array. For purposes of illustration, the local distortion values $d_{i,j}$ are shown to exist throughout the array prior to being used in the computation, but it is clear that only distortion values on the diagonal performing computations are required, and hence that the process can be pipelined.

Each cell receives four path weight symbols (two from the cell below and two from the cell on the left) and, likewise, outputs four symbols. The generation of these quantities is shown in figure 4.17. As the computational front progresses, each computing diagonal has available all the required inputs and computes all outputs required by the next diagonal.

The complete process, the aggregate of local distortion computation, path computation and pipelining, is shown for two adjacent pairs of utterances in figure 4.18.

1A-71.962



1. $d_{ij} = \sum_{n=0}^{P} u_n^{\ell}(i)\, r_n^{\ell}(j)$

2. $\hat{C}_{i,j} = d_{ij}' + \min(C_{i-1,j-1}, C_{i-1,j-2})$

3. $C_{i,j} = \min(\hat{C}_{i,j}, d_{ij}' + \hat{C}_{i-1,j})$

Figure 4.16. DTW ARRAY PROCESSING ELEMENT

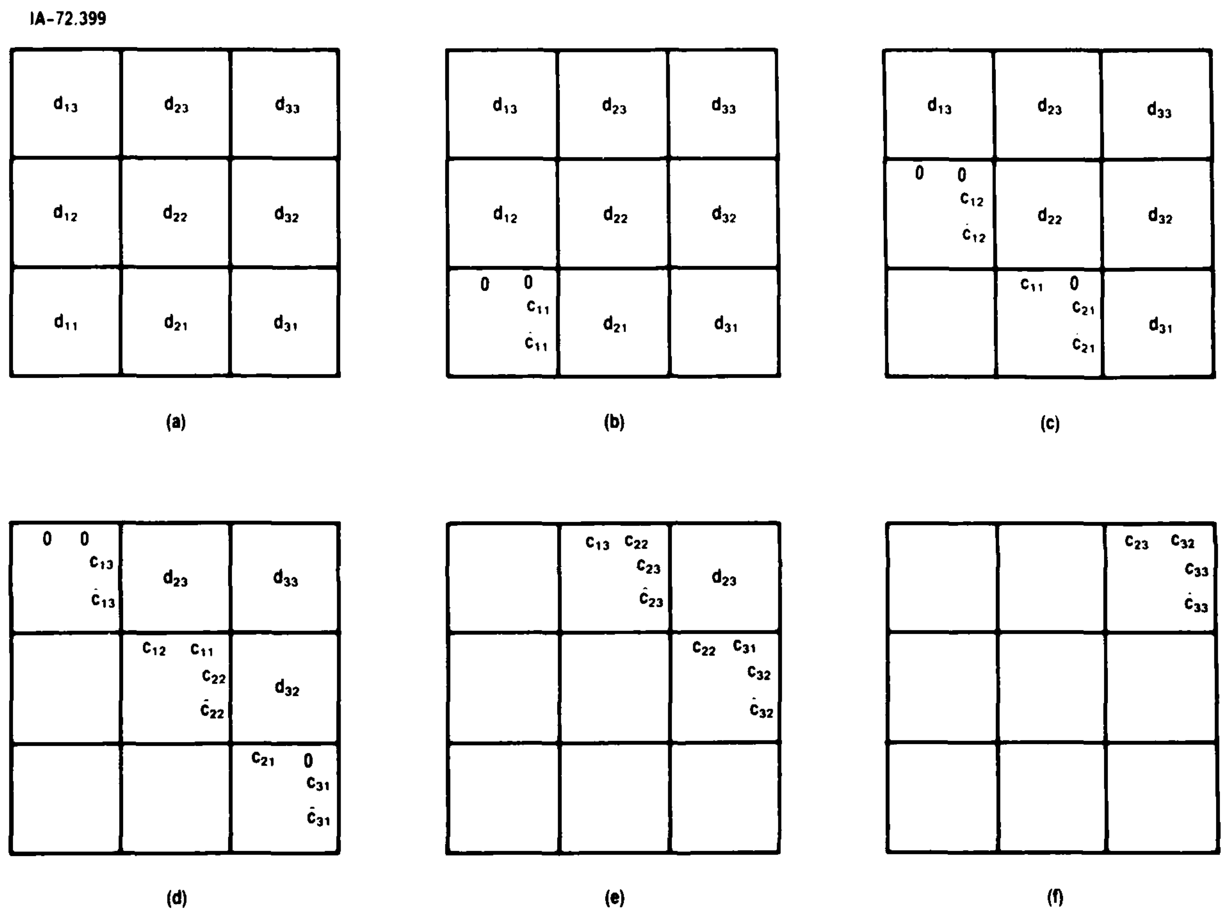
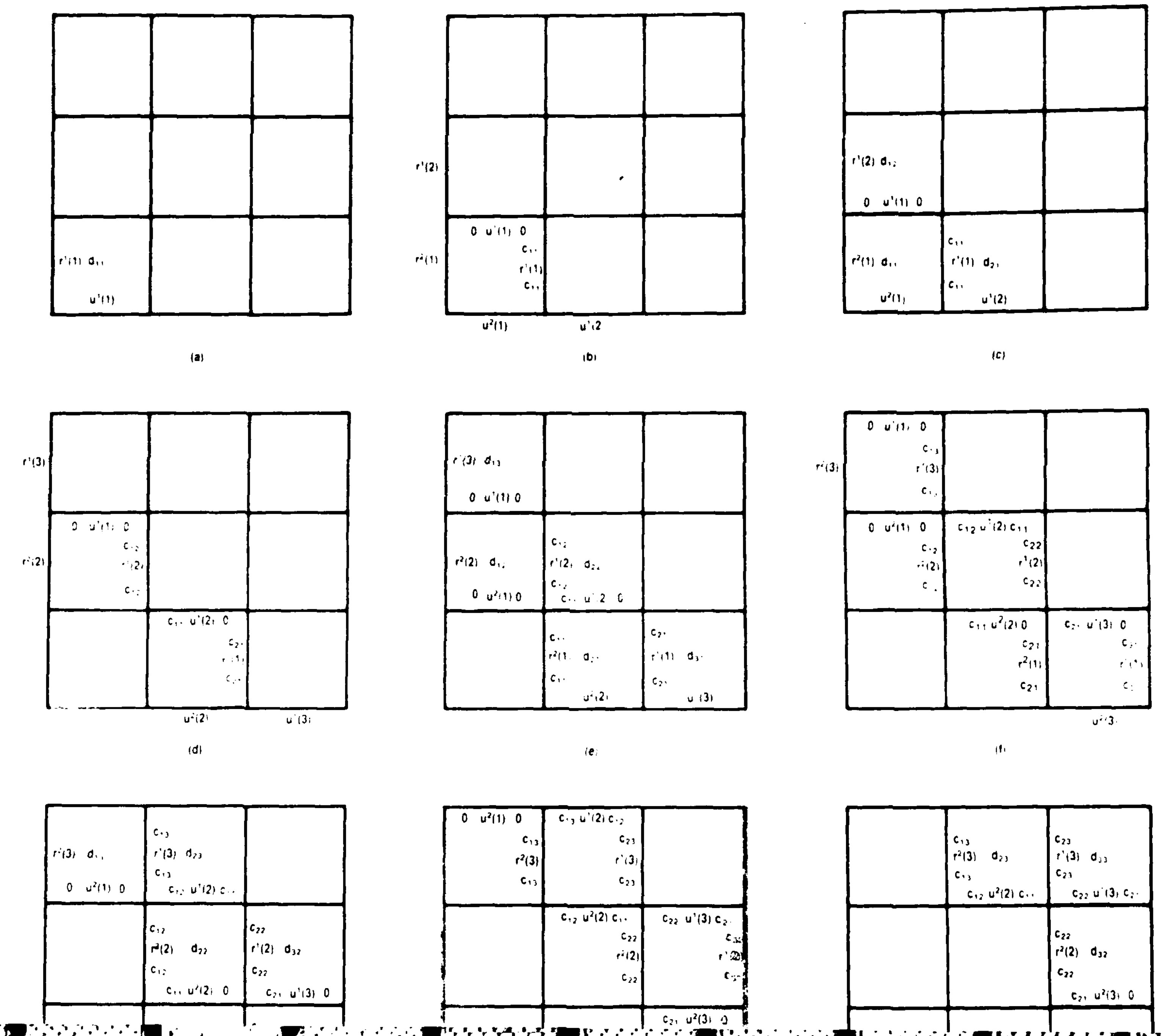

IA-72.399
(a) d_{13} d_{23} d_{33} d_{12} d_{22} d_{32} d_{11} d_{21} d_{31}
(b) d_{13} d_{23} d_{33} d_{12} d_{22} d_{32} 0 0 c_{11} $\dot{c}_{11}$ d_{21} d_{31}
(c) d_{13} d_{23} d_{33} 0 0 c_{12} $\dot{c}_{12}$ d_{22} d_{32} c_{11} 0 c_{21} $\dot{c}_{21}$ d_{31}
(d) 0 0 c_{13} $\dot{c}_{13}$ d_{23} d_{33} c_{12} c_{11} c_{22} $\dot{c}_{22}$ d_{32} c_{21} 0 c_{31} $\dot{c}_{31}$
(e) c_{13} c_{22} c_{23} $\dot{c}_{23}$ d_{23} c_{22} c_{31} c_{32} $\dot{c}_{32}$
(f) c_{23} c_{32} c_{33} $\dot{c}_{33}$

Figure 4.17. DTW: SYSTOLIC PATH COMPUTATION


(a)
(b)
(c)
(d)
(e)
(f)
r¹(1) d₁₁
u¹(1)
u²(1)
u¹(2)
r¹(2)
r²(1)
r¹(3)
r²(2)
r²(3)
u²(2)
u¹(3)
u²(3)

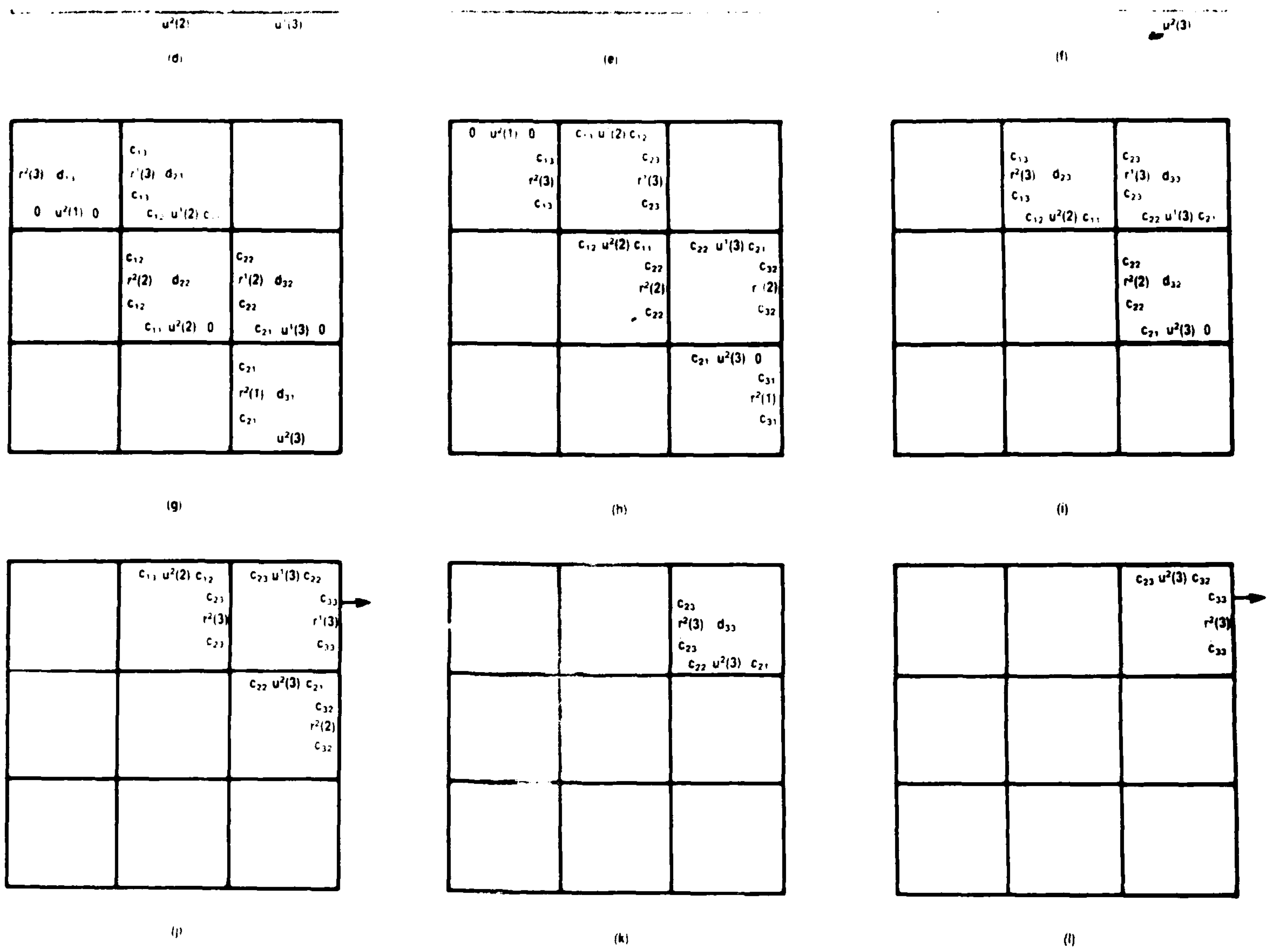


Figure 4.18. SYSTOLIC DTW COMPUTATION

The path computations are carried out in several residue channels, but the local path decisions, which require the computation of two minimums, are made using only one residue channel. The decisions are then communicated to the other channels.

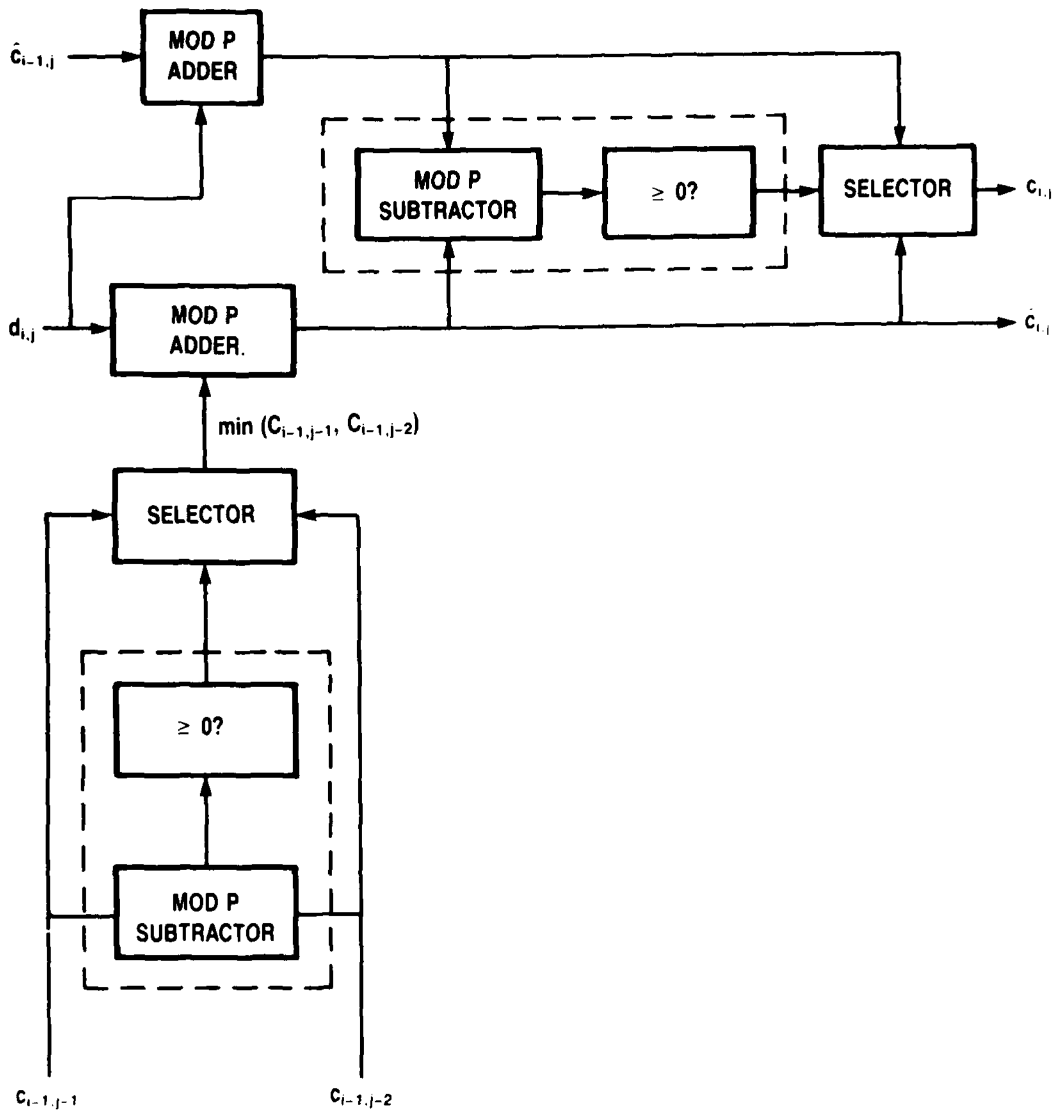
The required operations are

$$\hat{c}_{i,j} = d_{ij} + \min(c_{i-1,j-1}, c_{i-1,j-2})$$

$$c_{i,j} = \min(\hat{c}_{i,j}, d_{ij} + \hat{c}_{i-1,j}).$$

Each minimization involves one residue addition and two seven-bit table lookups. Computation of $c_{i,j}$ then takes $2\tau_s + 2\tau_t$ seconds. The sum indicated inside the parentheses in the second equation can be performed in parallel with the computation of $\hat{c}_{i,j}$, so that the computation of $c_{i,j}$ contributes $2\tau_s + 2\tau_t$ seconds.

A block diagram of the cell operations that follow the computation of d_{ij} is given in figure 4.19. The two dashed boxes contain the decision making portion of the minimizations, and only one set of these is required per cell. Each residue channel contains a set of selectors, which receive the two quantities being minimized, as well as the decision signal which carries one bit of information.



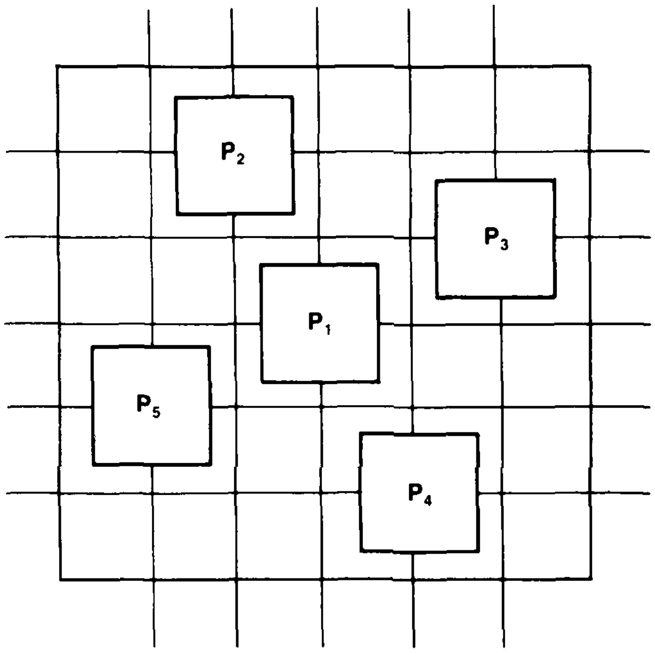
IA-72.396

Figure 4.19. DTW MOD P CELL ARCHITECTURE

4.3.3 Interconnection Concepts

A solution to the seemingly difficult wiring problem required by the communication between the (typically 50-by-50) arrays corresponding to the different residues is to define the basic cell so that it contains all the residues, as in figure 4.20. The wiring indicated in the figure could be carried on two levels, with all horizontal lines on one level and all vertical lines on the other one. All path computations could be performed in the central, and largest of all the cells.

All lines indicated are one-bit lines, with all data being transmitted serially at high speed. Each one of the five data lines on each side of the cell would be used to shift in 13 seven-bit symbols, with the central one shifting two additional seven-bit path cost symbols. This gives a total of 20 data lines, and, on any line, no more than 105 bits would have to be transmitted during one iteration. Assuming a 20 MHz one-bit shifting rate between one chip to another, this could be done in about 5 microseconds.



IA-71,958

Figure 4.20. RNS DTW CELL

4.3.4 Estimate of Throughput

The diagram in figure 4.21 shows the DTW cell operation on a time scale; the times indicated are based on the following table of estimated basic operation times.

7-bit residue adder	$\tau_s = 84$ ns
7-bit residue multiplier	$\tau_m = 300$ ns
7-bit input, 14-bit output table look-up (PLA	$\tau_t = 100$ ns
1-bit shift	$\tau_b = 50$ ns

The duration of the cell operating cycle is determined by the input data shift, which in turn is determined by the input-output pin limitation. If one cell is to occupy one VLSI IC, then two pins are required for VDD and ground, two pins for a two-phase clock, at least 20 pins for data and some more for testing. A 40-pin DIP could be used, which would have a few additional pins that could be used to speed up the data flow. An alternative is to put nine cells in a 3-by-3 array on an 84-pin PGA (Pin Grid Array), in which case the minimum number of required data lines would be 60. From figure 4.21 we see that the other operations can be performed in $13\tau_m$ and $17\tau_s$ seconds or about 5μs, so we conclude that one DTW output (one comparison between a pair of utterances) would be produced every 5μs. For a (fast) speaking rate of ten utterances per second, a 20,000 word library could be scanned for every utterance. The process could also be speeded up by doing more than one multiplication and addition in parallel when forming the scalar product of $\underline{u}(i)$ and $\underline{r}(j)$.

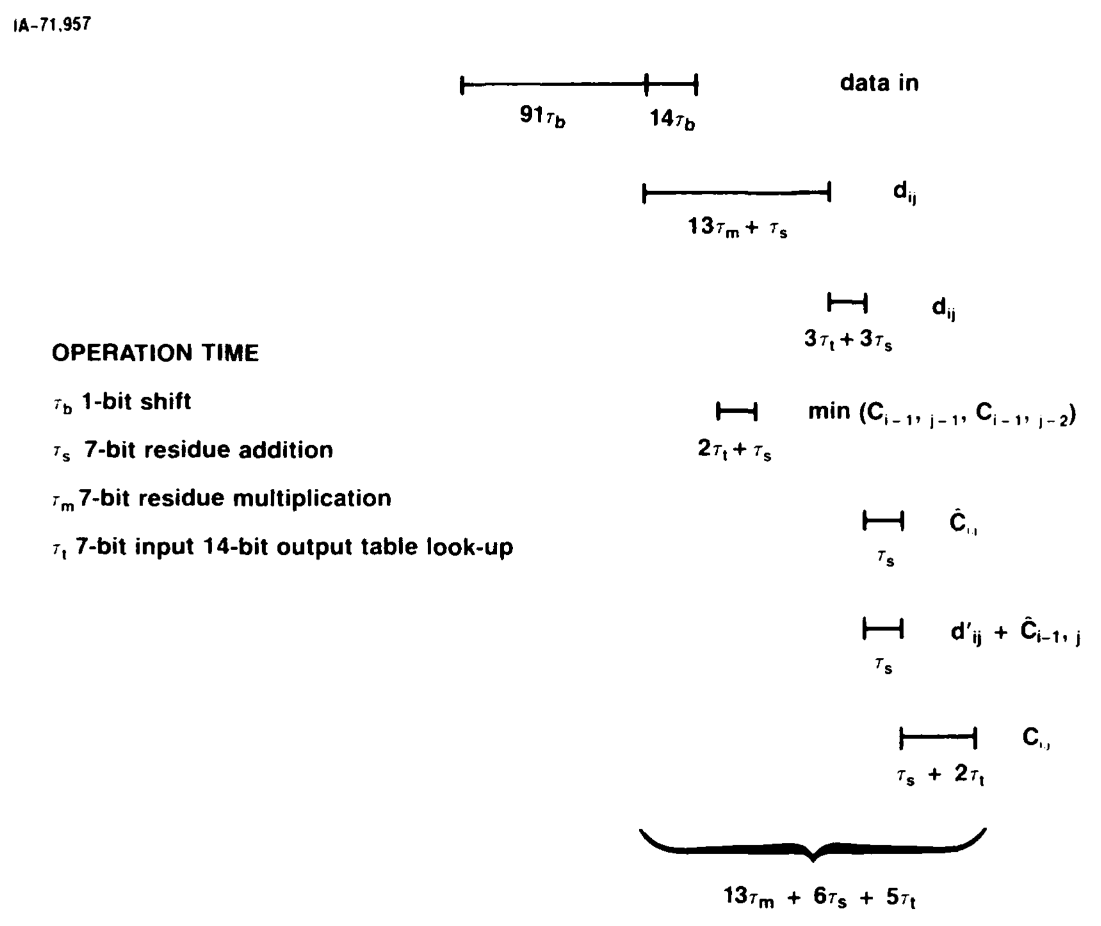

IA-71,957
data in
$91\tau_b$
$14\tau_b$
d_{ij}
$13\tau_m + \tau_s$
d_{ij}
$3\tau_t + 3\tau_s$
OPERATION TIME
τ_b 1-bit shift
τ_s 7-bit residue addition
τ_m 7-bit residue multiplication
τ_t 7-bit input 14-bit output table look-up
$\min(C_{i-1, j-1}, C_{i-1, j-2})$
$2\tau_t + \tau_s$
$\hat{C}_{ij}$
τ_s
$d'_{ij} + \hat{C}_{i-1, j}$
τ_s
C_{ij}
$\tau_s + 2\tau_t$
$13\tau_m + 6\tau_s + 5\tau_t$

Figure 4.21. SCHEDULE OF DTW CELL OPERATIONS

4.4 SUMMARY

Figure 4.22 depicts schematically a DTW array with pipelined inputs. The test inputs are provided by a bank of three correlators as discussed previously, while the reference vectors are taken from a stored library. The commutator sorts the auto‿orrelation vectors produced by the correlators, and the cells shown serve to provide the appropriate skew to the input data of the DTW array. It is assumed that the logarithms of the process gains (computed separately and converted to RNS form using the three moduli which provide the range for the DTW computation) are included with the correlation vectors for implementation of the Itakura-Saito metric. Finally, the DTW output must be converted to a weighted number system, possibly in mixed radix form before the different scores are normalized and compared.

IA-71 963

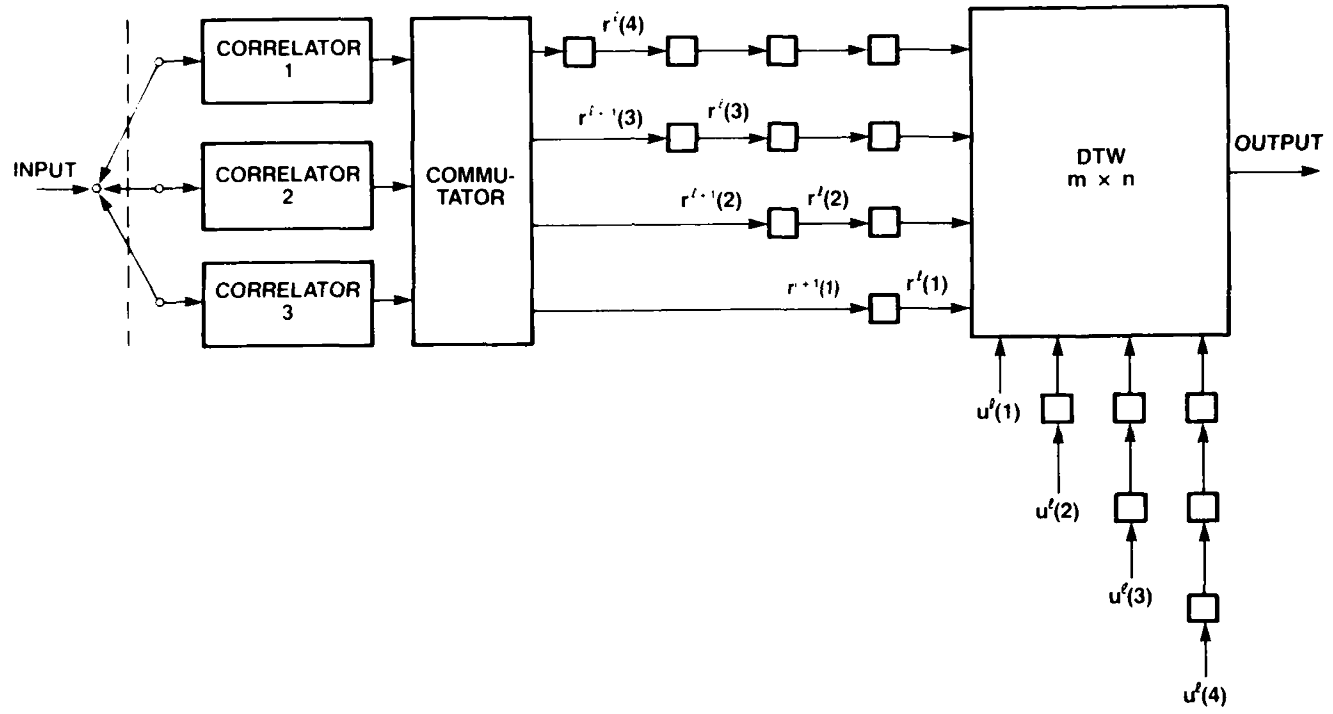


Figure 4.22. CORRELATOR AND DTW ARRAY

SECTION 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 SUMMARY

In the previous paragraphs we have reported on our work over the preceding year to demonstrate the effectiveness of the combination of RNS computations and systolic array architectures for the improved implementation of speech recognition algorithms, improvement being assessed with regard to throughput and complexity of hardware implementation. From our work we conclude that RNS does indeed have utility for implementing a dynamic time-warping word recognition algorithm driven by autoregressive spectral analysis and the Itakura-Saito distortion computation. We showed how the combination of RNS and systolic arrays could be used to effectively implement:

The autocorrelation estimates used in autoregressive (or LPC) spectral analysis;

Computation of the Itakura-Saito distortion values;

Dynamic time-warping least-cost path-metric computations.

The incorporation of the distortion computations with the path-metric computations in a pipelined two-dimensional systolic array was shown to have a favorable impact in providing a high throughput for the most computationally intensive part of the recognition algorithm.

These investigations were supported by the development of a computer simulation which was used extensively for experimentation with an eleven-word vocabulary, three to six different productions of each word being stored in a reference library to accommodate variations in utterance of the same word. From our experiments, we developed a means of reducing the dynamic range of the path-metric computations by quantization of the distortion values within RNS. A good means of narrowing the range of distortion values seems critical to successful RNS implementation.

A number of different distortion functions have been studied and reported in the speech processing literature; many of them are variants of the Itakura-Saito metric, and thus based on LPC analysis [19]. Experimental results have largely shown, in our opinion, that different metrics of this sort are roughly equivalent, although a ranking based on small improvements could be contemplated [3]. Much of the interest in LPC stems from its success in modeling the speech production process in a noise-free environment, but it is known that the all-zero linear prediction model breaks down in the presence of additive noise [19].

From our point of view, not only must the distortion measure produce satisfactory discrimination in a narrow range of values, it must also be suitable for RNS computations. For speech recognition, the purpose of the spectral analysis and distortion computation is to distill the information contained in the speech waveform into a small set of data suitable for low-error discrimination between distinct word patterns, and not to preserve information needed for high-grade speech synthesis. Thus, the LPC methods may be unnecessarily stringent for speech recognition purposes, while weak in the presence of noise, and at the same time imposing the need for high-precision, and therefore large dynamic-range, computations.

While we remain convinced by the results of our work that RNS implementation has high potential for speech recognition, we are also convinced that substantial improvement in reducing the gross dynamic range of the distortion computations, without sacrificing discrimination ability, is required before RNS implementation will be accepted as a truly practical or attractive alternative to conventional architectures that use floating-point computation. This will require a re-assessment of the distortion function used to support the DTW computations. Some alternative distortion function computations that could be considered in a follow-on effort are discussed briefly below.

5.2 ALTERNATIVES FOR DISTORTION COMPUTATION

In our work, we chose the LPC analysis and Itakura-Saito distortion for the reasons discussed in section 1, but perceive the need to experiment with other distortion functions in any follow-on effort. It is particularly perplexing that the path-metric computations can be successfully carried out with very coarse quantization of the distortion values, while our implementation of the Itakura-Saito metric shows a need to maintain a much larger range for its computation.

5.2.1 Itakura Distortion

One variation of the Itakura-Saito distortion function, sometimes called the gain-optimized Itakura-Saito distortion, was originally recommended by Itakura [21]. Its distinguishing feature is the use of a logarithmic function of the ratio of the linear prediction residuals. It provides a measure that is roughly proportional to the logarithm of the Itakura-Saito (or maximum likelihood) distortion. As such, it has the desirable attribute of compressing the range of distortion values. Hardware implementation of this method has been reported in the literature in which the logarithmic distortion values are confined to the range of a few bits and in which the path metric computations are carried out with 16-bit fixed-point arithmetic [10]. The key to that implementation is the simplicity of a logarithmic quantizer based on a geometric series representation of the logarithm in a form that is amenable to hardware implementation (with roughly 200 logic gates). The referenced work points out the feasibility of compressing the range of the Itakura-Saito distortion values (computed with 24-bit precision), but it is not directly of value to our RNS implementation. Since we would like to pipeline the distortion computations in the same RNS systolic array that is used for the path-metric computations, we would need to invent a logarithmic converter in RNS, and that is not a likely prospect.

5.2.2 Euclidean Distance Measures

Squared Euclidean distance, or equivalently mean-squared error, while traditional in signal processing work, has tended to be rejected in recent speech research on the subjective grounds that it is not sufficiently meaningful for representing what are thought to be the requirements of auditory perception. It is pointed out that the ear needs only to recognize the random process producing the waveform to within some accuracy and does not need to accurately reproduce the specific waveform, and that demanding a small mean-squared error in a speech system will often require far more bits and accuracy than the human ear requires [21]. The success of the LPC methods can be attributed to the corresponding distortion measures (such as Itakura-Saito´s) that measure in a probabilistic sense the closeness of the original and reproduced processes or models rather than the actual waveforms.

Mean-squared error, however, cannot be rejected on the grounds that it is too forgiving. If we base our analysis on power spectra, or equivalently autocorrelation analysis, then Euclidean distance may still be useful to discriminate spectral patterns for purposes of automatic speech recognition. For RNS implementation, we prefer squared Euclidean distance since it avoids the need for explicit sign detection which would suggest leaving RNS for that purpose. The utility of this measure will then depend on our ability to contain the range of values to a practical size. Two approaches that use squared Euclidean distance will be discussed briefly.

5.2.2.1 Log Spectral Deviation

One of the oldest distortion measures proposed for speech is formed by the L_p norm of the difference of the logarithms of the power spectra. Assuming the spectral envelopes have been sampled and scaled logarithmically, the L_2 norm is simply the squared Euclidean distance between the vectors of logarithmic spectral samples. One of the traditional ways of providing the spectral envelopes is by means of a bank of constant-Q filters appropriately spaced across the speech spectrum, the output of each filter's power being sampled in time and scaled logarithmically [22]. Such a filter bank is probably best implemented with analog-sampled switched-capacitor active filters rather than in the form of digital filters; thus we would not propose to use RNS for the filter bank, but would convert the log-spectral samples to RNS code for computation of the squared Euclidean distortion in a systolic array of the type described in section 4.

Another means of performing the filter-bank analysis would be to perform a discrete Fourier transform (DFT) of the windowed speech samples with a moderately high resolution (perhaps 256 to 1024 samples per frame), subdivide the samples into appropriate bands over which the complex DFT samples are to be squared and summed, compute the power in each sub-band and convert to logarithmic form. With the exception of logarithmic conversion, all of the processing could be carried out with RNS. Logarithmic conversion would require reconversion to a weighted number system (which would probably be needed for spectral normalization anyway) followed by reconversion to RNS for the distortion computation. These processing steps would clearly be more complicated than the computation of the autocorrelation samples described in section 4, but the operations would need to be performed only once for each test segment and the method should be worthy of consideration for future effort.

5.2.2.2 Direct Autocorrelation Analysis

Although the autocorrelation coefficients of the windowed speech samples could be transformed by DFT to provide a representation of the spectral envelope, it may be better to use them directly for spectral discrimination. As shown in section 2, all of the processing for estimating LPC coefficients and computing the Itakura-Saito distortion is carried out in the time domain, without any specific need for the frequency spectra. While LPC analysis is used to approximate the spectral envelope, as represented by the all-pole linear filter model, it is essentially a linear transformation of a subset of autocorrelation coefficients. The assumption of an all-pole model of the speech production process allows this subset of autocorrelation values to accurately approximate the remaining ones in accordance with the autoregressive nature of the model. This is the basis for obtaining a good spectral approximation.

If the LPC model is adequate, then for the purpose of automatic speech recognition it may be satisfactory to employ the subset of autocorrelation coefficients used in LPC analysis directly in a squared Euclidean distance computation. For RNS implementation, since it would be appropriate to work with normalized correlation coefficients, it would be necessary to leave RNS to carry out the normalization and then reconvert to RNS for the distance computations. But, such a technique for distortion computation might actually be simpler to implement than the Itakura-Saito distortion and there would be no need to solve the normal equations for construction of the reference library. Also, if the number of samples used is extended beyond the LPC subset, then the dependence on the assumption of an autoregressive model is diminished.

5.3 FOLLOW-ON RECOMMENDATIONS

In any follow-on effort we would propose to experimentally study the feasibility of using a squared-Euclidean metric, rather than the Itakura-Saito distortion, for RNS implementation of the distortion computations. The spectral envelope and direct autocorrelation methods discussed above would be a starting point for such investigations. A critical issue is the dynamic range imposed by the square-law measure, particularly when comparing dissimilar words. We must recognize, however, that occasional overflow of the computation range, in which RNS converts the statistical outliers to values lying in a valid range, is probably tolerated in the path metric computations, particularly since these events are more likely to occur while comparing dissimilar words whose best DTW path metrics should be relatively large in any case. It would be quite surprising if the overflows associated with bad word matches were to conspire to produce a path metric smaller than that of a proper match.

If either of the methods discussed were to prove viable in reducing the range of distortion values significantly, then it would be appropriate to contemplate actual VLSI hardware implementation of the simplified DTW systolic array, requiring an increased attention to the details of combinational logic design of the cells of the array in a follow-on effort.

Finally, the experimental data base should be expanded beyond the eleven-word vocabulary used in this study, and many controlled experiments performed, before definitive conclusions regarding RNS implementation and actual parameter selection can be made. These are the elements of a follow-on effort we expect to continue in FY 1985.

REFERENCES

1. Markel, J. D., and Gray, A. H., Linear Prediction of Speech, Springer-Verlag, 1976.

2. Itakura, F., and Saito, S., Analysis-Synthesis Telephony Based on the Maximum-Likelihood Method, Proceedings of 6th International Congr. Acoustics, Tokyo, Japan, 1968.

3. Gray, R. M., Buzo, A., Gray, A. H., Matsuyama, Y., Distortion Measures for Speech Processing, IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-28, vol. 4, August 1980.

4. Sakoe, H., and Chiba, S., Dynamic Programming Algorithm Optimization for Spoken Word Recognition, IEEE Trans. Acoustics, Speech, and Signal Processing ASSP-26, vol. 1, (1978), pp. 43-49.

5. Gray, R. M., On the Asymptotic Eigenvalue Distribution of Toeplitz Matrices, IEEE Transactions on Information Theory, IT-18, vol. 6, November 1972.

6. Itakura, F., and Saito, S., A Statistical Method for Estimation of Speech Spectral Density and Formant Frequencies, Electronics and Communications in Japan, Vol. 53-A, No. 1, 1970.

7. Kullback, S., Information Theory and Statistics, John Wiley, 1959.

8. Cramer, H., Random Variables and Probability Distributions, 2nd Ed., Cambridge University Press, 1962.

9. Schweppe, F. C., Uncertain Dynamic Systems, Prentice Hall, 1973.

10. Brown, M. K., Thorkildsen, R., Oh, Y. H., Ali, S. S., The DTWP: An LPC-Based Dynamic Time-Warping Processor for Isolated Word Recognition, AT&T Bell Laboratories Technical Journal, vol. 63, no. 3, March 1984, pp. 441-456.

REFERENCES (Continued)

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

12. Bellman, R., Dynamic Programming, Princeton University Press, 1957.

13. Moore, E. F., The Shortest Path Through a Maze, Proceedings of an International Symposium on the Theory of Switching, vol. II, Harvard University Press (Cambridge, 1959), pp. 258-292.

14. Dantzig, G. B., Discrete-Variable Extremum Problems, Operations Research, vol. 5, (1957), pp. 266-270.

15. Bellman, R., On a Routing Problem, Quarterly of Applied Mathematics XIV, vol. 1, (1958), pp. 87-90.

16. Aiken, H. H., and Semon, W. L., Advanced Digital Computer Logic, WADC TR-59-472, Wright-Patterson Air Force Base, Ohio (1959).

17. Eastman, W. L., Sign Determination in a Modular Number System, Proceedings of a Harvard Symposium on Digital Computers and Their Applications, Harvard University Press (Cambridge, 1962), pp. 136-162.

18. Semon, W. L., Some Properties of A(ψ), Notes on Modular Number Systems, ASD TR-61-12, Wright-Patterson Air Force Base, Ohio (1961), pp. 55-57.

19. Y. Matsuyama, A. Buzo, R. M. Gray, Spectral Distortion Measures for Speech Compression, Stanford Electronics Laboratories, Technical Report No. 6504-3, April 1983.

20. J. S. Lim and A. V. Oppenheim, All-Pole Modeling of Degraded Speech, IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-26, No. 3, June 1978.

21. Itakura, F., Minimum Prediction Residual Principle Applied to Speech Recognition, IEEE Trans. Acoustics, Speech, and Signal Processing ASSP-23, vol. 1, (1975), pp. 67-72.

22. L. R. Rabiner and R. W. Schafer, Digital Processing of Speech Signals, Prentice-Hall, 1978.

23. E. L. Key, Digital Signal Processing with Residue Number Systems, Proceedings IEEE International Conference on Computer Design: VLSI in Computers, October 1983.

APPENDIX A

RESIDUE NUMBER SYSTEMS

An RNS with range $M = p_1 p_2 \cdots p_\ell$, where each p_k is a prime integer, called a modulus, is a number system represented by the integers in the interval $[0,M-1]$ in which addition and multiplication of two elements x and y is defined as the remainder on division by M of their sum and product, respectively. Under these operations, the usual sum or product of x and y is obtained, provided that the result lies in the same range $[0, M - 1]$. The main advantage afforded by RNS is that the complexity of the operations (measured by the size of a table required to implement the operations, for instance) is lower than that of the corresponding integer operations because of the ability to decompose the processor into a set of independent processors each performing operations modulo p_k [23].

In RNS an integer x in the interval $[0,M-1]$ is represented by its residues modulo the p_k, i.e.,

$$x = (x_1, x_2, \ldots, x_\ell) \tag{A-1}$$

where x_k equals the remainder obtained when x is divided by p_k; it follows that, for each k, $0 \leq x_k < p_k$. Every number between 0 and M - 1, inclusively, has a unique RNS representation, and every ℓ-vector $(x_1, x_2, \ldots, x_\ell)$ with $0 \leq x_k < p_k$ corresponds to a unique integer x with $0 \leq x < M$. Furthermore, the sum (product) of two integers x and y in the RNS can be obtained by adding (multiplying) corresponding components x_k and y_k, modulo the corresponding p_k.

In this manner, any operation requiring only addition and multiplication, the input and output ranges of which are known, can be implemented in RNS. Since intermediate overflow is harmless, provided that the final result is contained in $[0,M-1]$, it follows that the range of the system is solely determined by the range of the output.

Figure A-1 illustrates the architecture of a general RNS processor. After the input has been converted to RNS form, the function is implemented by an independent set of processors, each operating in a different residue channel. Usually, the RNS output is converted to a weighted number system, but linear operations could be continued in RNS representation.

A formula relating a positive integer $x \in [0, M-1]$ to its RNS representation $(x_1, x_2, \ldots, x_\ell)$, is given by

$$x = \sum_{i=1}^{\ell} x_i M_i M_i^{-1} \bmod M \qquad \text{(A-2)}$$

where $M_i = M/p_i$ and M_i^{-1} is the inverse of M_i modulo p_i, i.e., an integer M_i^{-1} in the range $[0, p_i-1]$ such that

$$M_i M_i^{-1} \equiv 1 \mod p_i \qquad \text{(A-3)}$$

IA-72,436

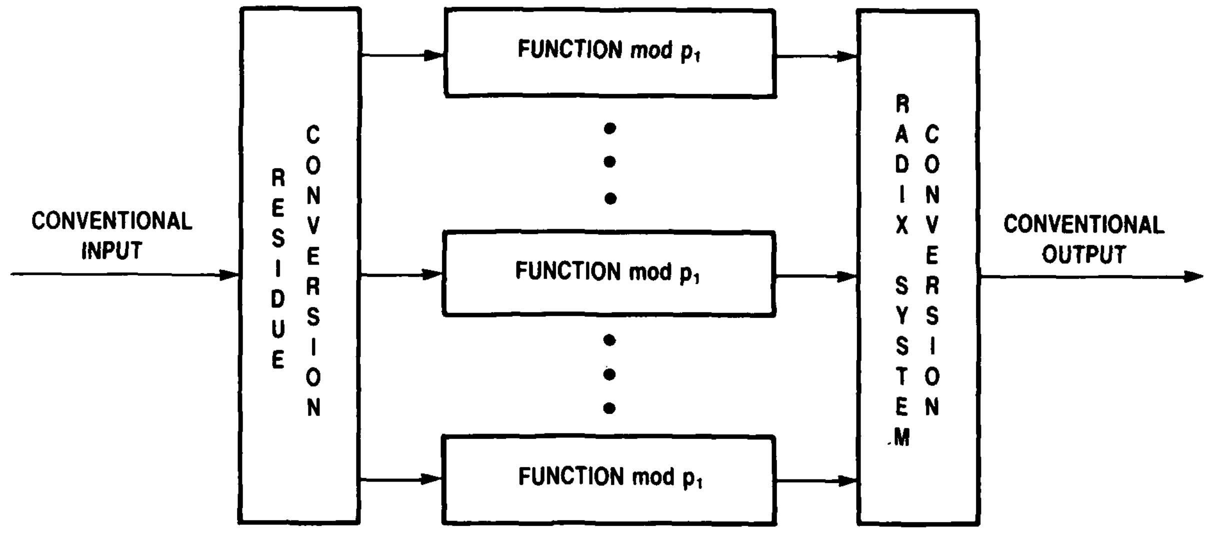


Figure A-1. GENERAL RESIDUE NUMBER SYSTEM FUNCTION

For example, if $p_1 = 3$, $p_2 = 5$, and $p_3 = 7$, then $M_1 = 35$, $M_2 = 21$, $M_3 = 15$, $M_1^{-1} = 2$, $M_2^{-1} = 1$, and $M_3^{-1} = 1$. Now, if $x = 34$, then $(x_1, x_2, x_3) = (1, 4, 6)$ and the formula gives

$$\begin{aligned} x &= 1 \cdot 35 \cdot 2 + 4 \cdot 21 \cdot 1 + 6 \cdot 15 \cdot 1 \mod 105 \\ &= 244 \mod 105 \qquad \text{(A-4)} \\ &= 34 \mod 105. \end{aligned}$$

This method of reconstruction, illustrated by (A-4), which is an example of the Chinese Remainder Theorem, is useful for theoretical purposes. A more efficient method for obtaining a weighted number representation of an RNS-coded number is to convert to a mixed-radix representation using the moduli of the RNS as radices.

The mixed-radix representation of an integer x in [0, M-1] is given by

$$x = \sum_{i=1}^{\ell} c_i \prod_{j=1}^{i-1} p_j \qquad \text{(A-5)}$$

where the c_i, called the mixed-radix coefficients, satisfy $0 \le c_i < p_i$. In equation (A-5),

$$1 \triangleq \prod_{j=1}^{0} p_j \qquad \text{(A-6)}$$

is [illegible]

For a three-moduli RNS, equation (A-5) can be written as

$$x = c_3 p_2 p_1 + c_2 p_1 + c_1. \tag{A-7}$$

Note that from equation (A-3) we have

$$\begin{aligned} c_1 &\equiv x \mod p_1 \\ c_2 &\equiv (x-c_1)/p_1 \mod p_2 \\ c_3 &\equiv \left(x - (x-c_1)/p_1\right)/p_2 \mod p_3 \end{aligned} \tag{A-8}$$

Equation (A-8) can be generalized to obtain the mixed-radix coefficients of a given number in an RNS with an arbitrary number of moduli. A refinement of the resulting equations yields

$$\begin{aligned} c_1 &\equiv x_1 \mod p_1 \\ c_2 &\equiv p_1^{-1}(x_2 - c_1) \mod p_2 \\ c_3 &\equiv p_2^{-1}\left(p_1^{-1}(x_3 - c_1) - c_2\right) \mod p_3 \\ &\vdots \\ c_\ell &\equiv p_{\ell-1}^{-1}(p_{\ell-2}^{-1}(\ldots p_1(x_\ell - c_1)\ldots - c_{\ell-2}) - c_{\ell-1}) \mod p_\ell \end{aligned} \tag{A-9}$$

The coefficients c_i are derived iteratively by starting with $c_1 = x_1$ and computing c_i by subtracting c_{i-1} from the result of the previous iteration and multiplying the difference by p_{i-1}^{-1}, all modulo p_i.

Finally, the evaluation of equation (A-5) can be done recursively using Horner's scheme:

$$x = (\cdots ((c_{\ell} p_{\ell-1} + c_{\ell-1}) p_{\ell-2} + c_{\ell-2}) p_{\ell-3} \cdots + c_2) p_1 + c_1 \quad \text{(A-10)}$$

Efficient VLSI hardware structures that perform pipelined binary-to-residue and residue-to-binary conversion for a five-modulus RNS have been designed by MITRE's Integrated Electronics project for the primes (31,29,23,19,17) representable with five bits.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

END
FILMED

5-86

DTIC