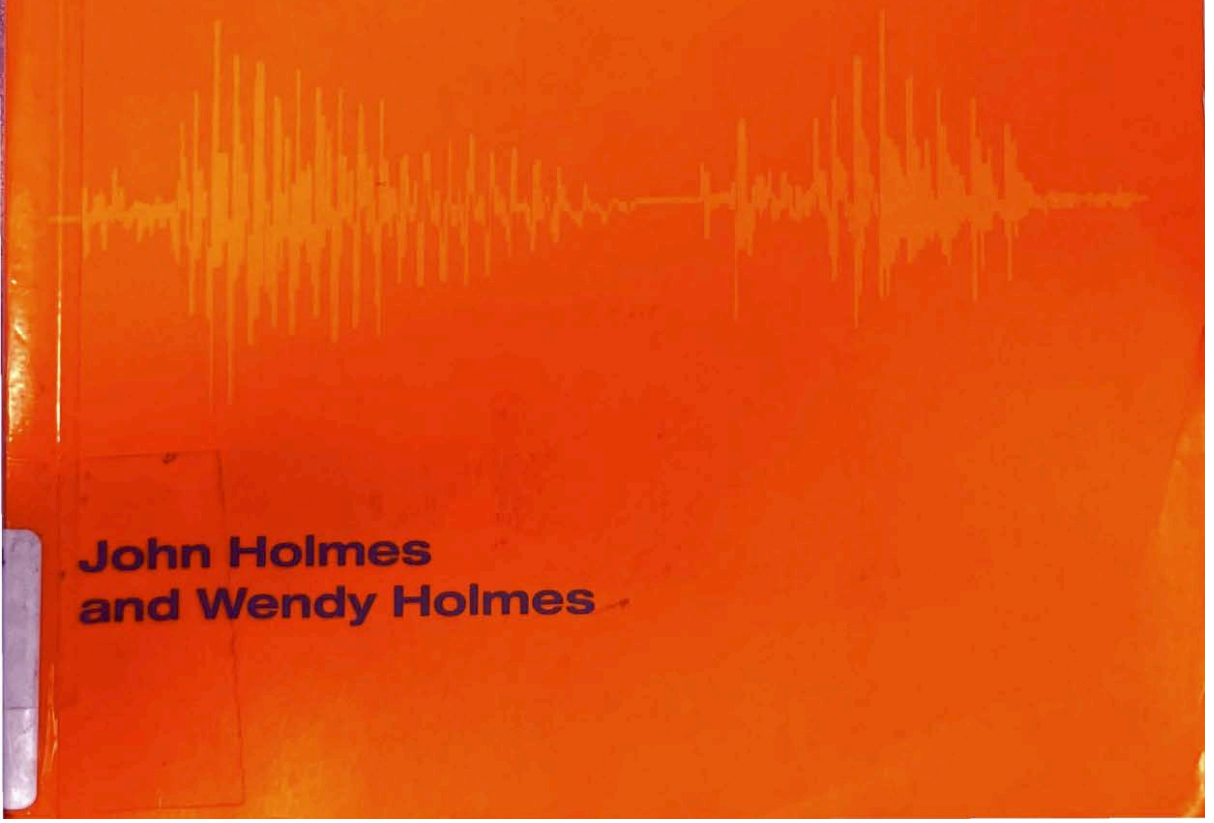
SACRAMENTO PUBLIC LIBRARY
3 3029 04671 7492

SPEECH SYNTHESIS AND RECOGNITION

2ND EDITION



John Holmes
and Wendy Holmes

Speech Synthesis and Recognition

Second Edition

John Holmes and Wendy Holmes



London and New York

Ex.1019 / Page 2 of 102
AMAZON.COM, INC.

First edition by the late Dr J.N. Holmes published 1988 by Van Nostrand Reinhold
Second edition published 2001 by Taylor & Francis
11 New Fetter Lane, London EC4P 4EE

Simultaneously published in the USA and Canada
by Taylor & Francis
29 West 35th Street, New York, NY 10001

Taylor & Francis is an imprint of the Taylor & Francis Group

© 2001 Wendy J. Holmes

Publisher's Note
This book has been prepared from camera-ready copy provided by the authors.
Printed and bound in Great Britain by Biddles Ltd, Guildford and King's Lynn

All rights reserved. No part of this book may be reprinted or reproduced or utilised in any form or by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying and recording, or in any information storage or retrieval system, without permission in writing from the publishers.

Every effort has been made to ensure that the advice and information in this book is true and accurate at the time of going to press. However, neither the publisher nor the authors can accept any legal responsibility or liability for any errors or omissions that may be made. In the case of drug administration, any medical procedure or the use of technical equipment mentioned within this book, you are strongly advised to consult the manufacturer's guidelines.

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Cataloging in Publication Data

Holmes, J.N.
Speech synthesis and recognition/John Holmes and Wendy Holmes.--2nd ed
p.cm.
Includes bibliographical references and index.
ISBN 0-7484-0856-8 (hc.) -- ISBN 0-7484-0857-6 (pbk.)
1. Speech processing systems. I. Holmes, Wendy (Wendy J.) II. Title.

TK77882.S65 H64 2002
006.4'54--dc21

2001044279

ISBN 0-7484-0856-8 (hbk)
ISBN 0-7484-0857-6 (pbk)

CONTENTS

Preface to the First Edition xiii
Preface to the Second Edition xv
List of Abbreviations xvii

1 Human Speech Communication 1

1.1 Value of speech for human–machine communication 1
1.2 Ideas and language 1
1.3 Relationship between written and spoken language 1
1.4 Phonetics and phonology 2
1.5 The acoustic signal 2
1.6 Phonemes, phones and allophones 3
1.7 Vowels, consonants and syllables 4
1.8 Phonemes and spelling 6
1.9 Prosodic features 6
1.10 Language, accent and dialect 7
1.11 Supplementing the acoustic signal 8
1.12 The complexity of speech processing 9
Chapter 1 summary 10
Chapter 1 exercises 10

2 Mechanisms and Models of Human Speech Production 11

2.1 Introduction 11
2.2 Sound sources 12
2.3 The resonant system 15
2.4 Interaction of laryngeal and vocal tract functions 19
2.5 Radiation 21
2.6 Waveforms and spectrograms 21
2.7 Speech production models 25
2.7.1 Excitation models 26
2.7.2 Vocal tract models 27
Chapter 2 summary 31
Chapter 2 exercises 32

3 Mechanisms and Models of the Human Auditory System 33

3.1 Introduction 33
3.2 Physiology of the outer and middle ears 33
3.3 Structure of the cochlea 34

Ex.1019 / Page 4 of 102
AMAZON.COM, INC.

3.4 Neural response 36
3.5 Psychophysical measurements 38
3.6 Analysis of simple and complex signals 41
3.7 Models of the auditory system 42
3.7.1 Mechanical filtering 42
3.7.2 Models of neural transduction 43
3.7.3 Higher-level neural processing 43
Chapter 3 summary 46
Chapter 3 exercises 46

4 Digital Coding of Speech 47

4.1 Introduction 47
4.2 Simple waveform coders 48
4.2.1 Pulse code modulation 48
4.2.2 Deltamodulation 50
4.3 Analysis/synthesis systems (vocoders) 52
4.3.1 Channel vocoders 53
4.3.2 Sinusoidal coders 53
4.3.3 LPC vocoders 54
4.3.4 Formant vocoders 56
4.3.5 Efficient parameter coding 57
4.3.6 Vocoders based on segmental/phonetic structure 58
4.4 Intermediate systems 58
4.4.1 Sub-band coding 59
4.4.2 Linear prediction with simple coding of the residual 60
4.4.3 Adaptive predictive coding 60
4.4.4 Multipulse LPC 62
4.4.5 Code-excited linear prediction 62
4.5 Evaluating speech coding algorithms 63
4.5.1 Subjective speech intelligibility measures 64
4.5.2 Subjective speech quality measures 64
4.5.3 Objective speech quality measures 64
4.6 Choosing a coder 65
Chapter 4 summary 66
Chapter 4 exercises 66

5 Message Synthesis from Stored Human Speech Components 67

5.1 Introduction 67
5.2 Concatenation of whole words 67
5.2.1 Simple waveform concatenation 67
5.2.2 Concatenation of vocoded words 70
5.2.3 Limitations of concatenating word-size units 71

Ex.1019 / Page 5 of 102
AMAZON.COM, INC.

5.3 Concatenation of sub-word units: general principles 71
5.3.1 Choice of sub-word unit 71
5.3.2 Recording and selecting data for the units 72
5.3.3 Varying durations of concatenative units 73
5.4 Synthesis by concatenating vocoded sub-word units 74
5.5 Synthesis by concatenating waveform segments 74
5.5.1 Pitch modification 75
5.5.2 Timing modification 77
5.5.3 Performance of waveform concatenation 77
5.6 Variants of concatenative waveform synthesis 78
5.7 Hardware requirements 79
Chapter 5 summary 80
Chapter 5 exercises 80

6 Phonetic synthesis by rule 81

6.1 Introduction 81
6.2 Acoustic-phonetic rules 81
6.3 Rules for formant synthesizers 82
6.4 Table-driven phonetic rules 83
6.4.1 Simple transition calculation 84
6.4.2 Overlapping transitions 85
6.4.3 Using the tables to generate utterances 86
6.5 Optimizing phonetic rules 89
6.5.1 Automatic adjustment of phonetic rules 89
6.5.2 Rules for different speaker types 90
6.5.3 Incorporating intensity rules 91
6.6 Current capabilities of phonetic synthesis by rule 91
Chapter 6 summary 92
Chapter 6 exercises 92

7 Speech Synthesis from Textual or Conceptual Input 93

7.1 Introduction 93
7.2 Emulating the human speaking process 93
7.3 Converting from text to speech 94
7.3.1 TTS system architecture 94
7.3.2 Overview of tasks required for TTS conversion 96
7.4 Text analysis 97
7.4.1 Text pre-processing 97
7.4.2 Morphological analysis 99
7.4.3 Phonetic transcription 100
7.4.4 Syntactic analysis and prosodic phrasing 101
7.4.5 Assignment of lexical stress and pattern of word accents 102

Ex.1019 / Page 6 of 102
AMAZON.COM, INC.

7.5 Prosody generation 102
7.5.1 Timing pattern 103
7.5.2 Fundamental frequency contour 104
7.6 Implementation issues 106
7.7 Current TTS synthesis capabilities 107
7.8 Speech synthesis from concept 107
Chapter 7 summary 108
Chapter 7 exercises 108

8 Introduction to automatic speech recognition: template matching 109

8.1 Introduction 109
8.2 General principles of pattern matching 109
8.3 Distance metrics 110
8.3.1 Filter-bank analysis 111
8.3.2 Level normalization 112
8.4 End-point detection for isolated words 114
8.5 Allowing for timescale variations 115
8.6 Dynamic programming for time alignment 115
8.7 Refinements to isolated-word DP matching 117
8.8 Score pruning 118
8.9 Allowing for end-point errors 121
8.10 Dynamic programming for connected words 121
8.11 Continuous speech recognition 124
8.12 Syntactic constraints 125
8.13 Training a whole-word recognizer 125
Chapter 8 summary 126
Chapter 8 exercises 126

9 Introduction to stochastic modelling 127

9.1 Feature variability in pattern matching 127
9.2 Introduction to hidden Markov models 128
9.3 Probability calculations in hidden Markov models 130
9.4 The Viterbi algorithm 133
9.5 Parameter estimation for hidden Markov models 134
9.5.1 Forward and backward probabilities 135
9.5.2 Parameter re-estimation with forward and backward probabilities 136
9.5.3 Viterbi training 139
9.6 Vector quantization 140
9.7 Multi-variate continuous distributions 141
9.8 Use of normal distributions with HMMs 142
9.8.1 Probability calculations 143
9.8.2 Estimating the parameters of a normal distribution 144

Ex.1019 / Page 7 of 102
AMAZON.COM, INC.

9.8.3 Baum–Welch re-estimation 144
9.8.4 Viterbi training 145
9.9 Model initialization 146
9.10 Gaussian mixtures 147
9.10.1 Calculating emission probabilities 147
9.10.2 Baum–Welch re-estimation 148
9.10.3 Re-estimation using the most likely state sequence 149
9.10.4 Initialization of Gaussian mixture distributions 150
9.10.5 Tied mixture distributions 151
9.11 Extension of stochastic models to word sequences 152
9.12 Implementing probability calculations 153
9.12.1 Using the Viterbi algorithm with probabilities in logarithmic form 153
9.12.2 Adding probabilities when they are in logarithmic form 154
9.13 Relationship between DTW and a simple HMM 155
9.14 State durational characteristics of HMMs 156
Chapter 9 summary 157
Chapter 9 exercises 158

10 Introduction to front-end analysis for automatic speech recognition 159

10.1 Introduction 159
10.2 Pre-emphasis 159
10.3 Frames and windowing 159
10.4 Filter banks, Fourier analysis and the mel scale 160
10.5 Cepstral analysis 161
10.6 Analysis based on linear prediction 165
10.7 Dynamic features 166
10.8 Capturing the perceptually relevant information 167
10.9 General feature transformations 167
10.10 Variable-frame-rate analysis 167
Chapter 10 summary 168
Chapter 10 exercises 168

11 Practical techniques for improving speech recognition performance 169

11.1 Introduction 169
11.2 Robustness to environment and channel effects 169
11.2.1 Feature-based techniques 171
11.2.2 Model-based techniques 171
11.2.3 Dealing with unknown or unpredictable noise corruption 173
11.3 Speaker-independent recognition 174
11.3.1 Speaker normalization 175
11.4 Model adaptation 176

Ex.1019 / Page 8 of 102
AMAZON.COM, INC.

11.4.1 Bayesian methods for training and adaptation of HMMs 176
11.4.2 Adaptation methods based on linear transforms 178
11.5 Discriminative training methods 179
11.5.1 Maximum mutual information training 179
11.5.2 Training criteria based on reducing recognition errors 180
11.6 Robustness of recognizers to vocabulary variation 181
Chapter 11 summary 181
Chapter 11 exercises 182

12 Automatic speech recognition for large vocabularies 183

12.1 Introduction 183
12.2 Historical perspective 183
12.3 Speech transcription and speech understanding 184
12.4 Speech transcription 185
12.5 Challenges posed by large vocabularies 186
12.6 Acoustic modelling 187
12.6.1 Context-dependent phone modelling 188
12.6.2 Training issues for context-dependent models 188
12.6.3 Parameter tying 190
12.6.4 Training procedure 190
12.6.5 Methods for clustering model parameters 193
12.6.6 Constructing phonetic decision trees 194
12.6.7 Extensions beyond triphone modelling 195
12.7 Language modelling 196
12.7.1 *N*-grams 197
12.7.2 Perplexity and evaluating language models 197
12.7.3 Data sparsity in language modelling 198
12.7.4 Discounting 199
12.7.5 Backing off in language modelling 200
12.7.6 Interpolation of language models 200
12.7.7 Choice of more general distribution for smoothing 201
12.7.8 Improving on simple *N*-grams 202
12.8 Decoding 203
12.8.1 Efficient one-pass Viterbi decoding for large vocabularies 203
12.8.2 Multiple-pass Viterbi decoding 204
12.8.3 Depth-first decoding 205
12.9 Evaluating LVCSR performance 205
12.9.1 Measuring errors 205
12.9.2 Controlling word insertion errors 206
12.9.3 Performance evaluations 206
12.10 Speech understanding 209
12.10.1 Measuring and evaluating speech understanding performance 210
Chapter 12 summary 211
Chapter 12 exercises 212

Ex.1019 / Page 9 of 102
AMAZON.COM, INC.

13 Neural networks for speech recognition 213

13.1 Introduction 213
13.2 The human brain 213
13.3 Connectionist models 214
13.4 Properties of ANNs 215
13.5 ANNs for speech recognition 216
13.5.1 Hybrid HMM/ANN methods 217
Chapter 13 summary 218
Chapter 13 exercises 218

14 Recognition of speaker characteristics 219

14.1 Characteristics of speakers 219
14.2 Verification versus identification 219
14.2.1 Assessing performance 220
14.2.2 Measures of verification performance 221
14.3 Speaker recognition 224
14.3.1 Text dependence 224
14.3.2 Methods for text-dependent/text-prompted speaker recognition 224
14.3.3 Methods for text-independent speaker recognition 225
14.3.4 Acoustic features for speaker recognition 226
14.3.5 Evaluations of speaker recognition performance 227
14.4 Language recognition 228
14.4.1 Techniques for language recognition 228
14.4.2 Acoustic features for language recognition 229
Chapter 14 summary 230
Chapter 14 exercises 230

15 Applications and performance of current technology 231

15.1 Introduction 231
15.2 Why use speech technology? 231
15.3 Speech synthesis technology 232
15.4 Examples of speech synthesis applications 233
15.4.1 Aids for the disabled 233
15.4.2 Spoken warning signals, instructions and user feedback 233
15.4.3 Education, toys and games 234
15.4.4 Telecommunications 234
15.5 Speech recognition technology 235
15.5.1 Characterizing speech recognizers and recognition tasks 235
15.5.2 Typical recognition performance for different tasks 237
15.5.3 Achieving success with ASR in an application 238
15.6 Examples of ASR applications 239

Ex.1019 Page 10 of 102
AMAZON.COM, INC.

15.6.1 Command and control 239
15.6.2 Education, toys and games 239
15.6.3 Dictation 240
15.6.4 Data entry and retrieval 240
15.6.5 Telecommunications 241
15.7 Applications of speaker and language recognition 243
15.8 The future of speech technology applications 243
Chapter 15 summary 244
Chapter 15 exercises 244

16 Future research directions in speech synthesis and recognition 245

16.1 Introduction 245
16.2 Speech synthesis 245
16.2.1 Speech sound generation 246
16.2.2 Prosody generation and higher-level linguistic processing 247
16.3 Automatic speech recognition 248
16.3.1 Advantages of statistical pattern-matching methods 248
16.3.2 Limitations of HMMs for speech recognition 249
16.3.3 Developing improved recognition models 250
16.4 Relationship between synthesis and recognition 252
16.5 Automatic speech understanding 253
Chapter 16 summary 254
Chapter 16 exercises 254

17 Further Reading 255

17.1 Books 255
17.2 Journals 256
17.3 Conferences and workshops 256
17.4 The Internet 257
17.5 Reading for individual chapters 258

References 265
Solutions to Exercises 277
Glossary 283
Index 287

Ex.1019 / Page 11 of 102
AMAZON.COM, INC.

CHAPTER 8

Introduction to Automatic Speech Recognition: Template Matching

8.1 INTRODUCTION

Much of the early work on **automatic speech recognition** (**ASR**), starting in the 1950s, involved attempting to apply rules based either on acoustic/phonetic knowledge or in many cases on simple *ad hoc* measurements of properties of the speech signal for different types of speech sound. The intention was to decode the signal directly into a sequence of phoneme-like units. These early methods, extensively reviewed by Hyde (1972), achieved very little success. The poor results were mainly because co-articulation causes the acoustic properties of individual phones to vary very widely, and any rule-based hard decisions about phone identity will often be wrong if they use only local information. Once wrong decisions have been made at an early stage, it is extremely difficult to recover from the errors later.

An alternative to rule-based methods is to use **pattern-matching** techniques. Primitive pattern-matching approaches were being investigated at around the same time as the early rule-based methods, but major improvements in speech recognizer performance did not occur until more general pattern-matching techniques were invented. This chapter describes typical methods that were developed for spoken word recognition during the 1970s. Although these methods were widely used in commercial speech recognizers in the 1970s and 1980s, they have now been largely superseded by more powerful methods (to be described in later chapters), which can be understood as a generalization of the simpler pattern-matching techniques introduced here. A thorough understanding of the principles of the first successful pattern-matching methods is thus a valuable introduction to the later techniques.

8.2 GENERAL PRINCIPLES OF PATTERN MATCHING

When a person utters a word, as we saw in Chapter 1, the word can be considered as a sequence of phonemes (the linguistic units) and the phonemes will be realized as phones. Because of inevitable co-articulation, the acoustic patterns associated with individual phones overlap in time, and therefore depend on the identities of their neighbours. Even for a word spoken in isolation, therefore, the acoustic pattern is related in a very complicated way to the word's linguistic structure.

However, if the same person repeats the same isolated word on separate occasions, the pattern is likely to be generally similar, because the same phonetic relationships will apply. Of course, there will probably also be differences, arising from many causes. For example, the second occurrence might be spoken faster or more slowly; there may be differences in vocal effort; the pitch and its variation during the word could be different; one example may be spoken more precisely

than the other, etc. It is obvious that the waveform of separate utterances of the same word may be very different. There are likely to be more similarities between spectrograms because (assuming that a short time-window is used, see Section 2.6), they better illustrate the vocal-tract resonances, which are closely related to the positions of the articulators. But even spectrograms will differ in detail due to the above types of difference, and timescale differences will be particularly obvious.

A well-established approach to ASR is to store in the machine example acoustic patterns (called **templates**) for all the words to be recognized, usually spoken by the person who will subsequently use the machine. Any incoming word can then be compared in turn with all words in the store, and the one that is most similar is assumed to be the correct one. In general none of the templates will match perfectly, so to be successful this technique must rely on the correct word being more similar to its own template than to any of the alternatives.

It is obvious that in some sense the sound pattern of the correct word is likely to be a better match than a wrong word, because it is made by more similar articulatory movements. Exploiting this similarity is, however, critically dependent on how the word patterns are compared, i.e. on how the 'distance' between two word examples is calculated. For example, it would be useless to compare waveforms, because even very similar repetitions of a word will differ appreciably in waveform detail from moment to moment, largely due to the difficulty of repeating the intonation and timing exactly.

It is implicit in the above comments that it must also be possible to identify the start and end points of words that are to be compared.

8.3 DISTANCE METRICS

In this section we will consider the problem of comparing the templates with the incoming speech when we know that corresponding points in time will be associated with similar articulatory events. In effect, we appear to be assuming that the words to be compared are spoken in isolation at exactly the same speed, and that their start and end points can be reliably determined. In practice these assumptions will very rarely be justified, and methods of dealing with the resultant problems will be discussed later in the chapter.

In calculating a distance between two words it is usual to derive a short-term distance that is local to corresponding parts of the words, and to integrate this distance over the entire word duration. Parameters representing the acoustic signal must be derived over some span of time, during which the properties are assumed not to change much. In one such span of time the measurements can be stored as a set of numbers, or **feature vector**, which may be regarded as representing a point in multi-dimensional space. The properties of a whole word can then be described as a succession of feature vectors (often referred to as **frames**), each representing a time slice of, say, 10–20 ms. The integral of the distance between the patterns then reduces to a sum of distances between corresponding pairs of feature vectors. To be useful, the distance must not be sensitive to small differences in intensity between otherwise similar words, and it should not give too much weight to differences in pitch. Those features of the acoustic signal that are determined by the phonetic properties should obviously be given more weight in the distance calculation.

Ex.1019 / Page 13 of 102
AMAZON.COM, INC.

8.3.1 Filter-bank analysis

The most obvious approach in choosing a distance metric which has some of the desirable properties is to use some representation of the short-term power spectrum. It has been explained in Chapter 2 how the short-term spectrum can represent the effects of moving formants, excitation spectrum, etc.

Although in tone languages pitch needs to be taken into account, in Western languages there is normally only slight correlation between pitch variations and the phonetic content of a word. The likely idiosyncratic variations of pitch that will occur from occasion to occasion mean that, except for tone languages, it is normally safer to ignore pitch in whole-word pattern-matching recognizers. Even for tone languages it is probably desirable to analyse pitch variations separately from effects due to the vocal tract configuration. It is best, therefore, to make the bandwidth of the spectral resolution such that it will not resolve the harmonics of the fundamental of voiced speech. Because the excitation periodicity is evident in the amplitude variations of the output from a broad-band analysis, it is also necessary to apply some time-smoothing to remove it. Such time-smoothing will also remove most of the fluctuations that result from randomness in turbulent excitation.

At higher frequencies the precise formant positions become less significant, and the resolving power of the ear (**critical bandwidth** – see Chapter 3) is such that detailed spectral information is not available to human listeners at high frequencies. It is therefore permissible to make the spectral analysis less selective, such that the effective filter bandwidth is several times the typical harmonic spacing. The desired analysis can thus be provided by a set of bandpass filters whose bandwidths and

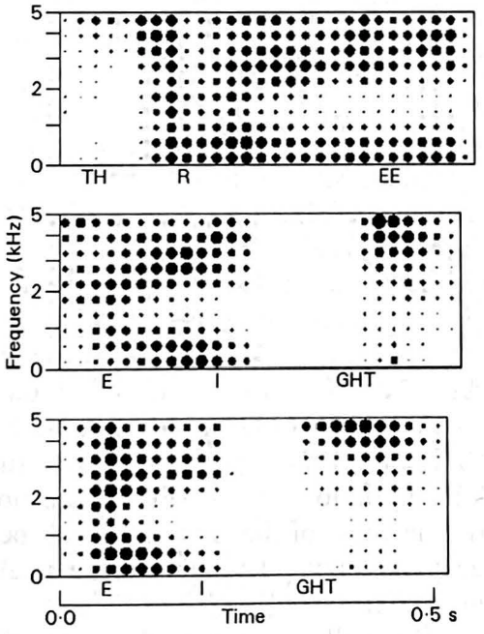


Figure 8.1 Spectrographic displays of a 10-channel filter-bank analysis (with a non-linear frequency spacing of the channels), shown for one example of the word "three" and two examples of the word "eight". It can be seen that the examples of "eight" are generally similar, although the lower one has a shorter gap for the [t] and a longer burst.

Ex.1019 / Page 14 of 102
AMAZON.COM, INC.

spacings are roughly equal to those of critical bands and whose range of centre frequencies covers the frequencies most important for speech perception (say from 300 Hz up to around 5 kHz). The total number of band-pass filters is therefore not likely to be more than about 20, and successful results have been achieved with as few as 10. When the necessary time-smoothing is included, the feature vector will represent the signal power in the filters averaged over the frame interval.

The usual name for this type of speech analysis is **filter-bank** analysis. Whether it is provided by a bank of discrete filters, implemented in analogue or digital form, or is implemented by sampling the outputs from short-term Fourier transforms, is a matter of engineering convenience. Figure 8.1 displays word patterns from a typical 10-channel filter-bank analyser for two examples of one word and one example of another. It can be seen from the frequency scales that the channels are closer together in the lower-frequency regions.

A consequence of removing the effect of the fundamental frequency and of using filters at least as wide as critical bands is to reduce the amount of information needed to describe a word pattern to much less than is needed for the waveform. Thus storage and computation in the pattern-matching process are much reduced.

8.3.2 Level normalization

Mean speech level normally varies by a few dB over periods of a few seconds, and changes in spacing between the microphone and the speaker's mouth can also cause changes of several dB. As these changes will be of no phonetic significance, it is desirable to minimize their effects on the distance metric. Use of filter-bank power directly gives most weight to more intense regions of the spectrum, where a change of 2 or 3 dB will represent a very large absolute difference. On the other hand, a 3 dB difference in one of the weaker formants might be of similar phonetic significance, but will cause a very small effect on the power. This difficulty can be avoided to a large extent by representing the power logarithmically, so that similar power ratios have the same effect on the distance calculation whether they occur in intense or weak spectral regions. Most of the phonetically unimportant variations discussed above will then have much less weight in the distance calculation than the differences in spectrum level that result from formant movements, etc.

Although comparing levels logarithmically is advantageous, care must be exercised in very low-level sounds, such as weak fricatives or during stop-consonant closures. At these times the logarithm of the level in a channel will depend more on the ambient background noise level than on the speech signal. If the speaker is in a very quiet environment the logarithmic level may suffer quite wide irrelevant variations as a result of breath noise or the rustle of clothing. One way of avoiding this difficulty is to add a small constant to the measured level before taking logarithms. The value of the constant would be chosen to dominate the greatest expected background noise level, but to be small compared with the level usually found during speech.

Differences in vocal effort will mainly have the effect of adding a constant to all components of the log spectrum, rather than changing the shape of the spectrum cross-section. Such differences can be made to have no effect on the distance metric by subtracting the mean of the logarithm of the spectrum level of each frame

Ex.1019 / Page 15 of 102
AMAZON.COM, INC.

from all the separate spectrum components for the frame. In practice this amount of level compensation is undesirable because extreme level variations are of some phonetic significance. For example, a substantial part of the acoustic difference between [f] and any vowel is the difference in level, which can be as much as

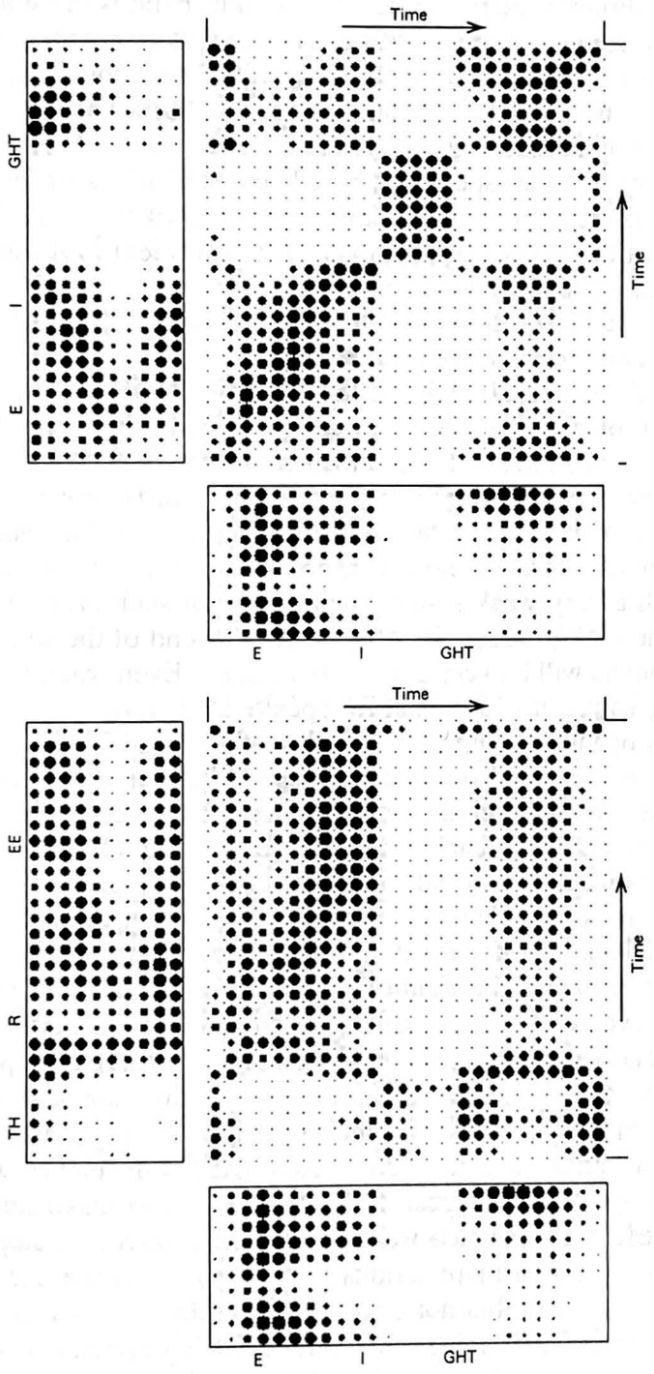


Figure 8.2 Graphical representation of the distance between frames of the spectrograms shown in Figure 8.1. The larger the blob the smaller the distance. It can be seen that there is a continuous path of fairly small distances between the bottom left and top right when the two examples of "eight" are compared, but not when "eight" is compared with "three".

30 dB. Recognition accuracy might well suffer if level differences of this magnitude were ignored. A useful compromise is to compensate only partly for level variations, by subtracting some fraction (say in the range 0.7 to 0.9) of the mean logarithmic level from each spectral channel. There are also several other techniques for achieving a similar effect.

A suitable distance metric for use with a filter bank is the sum of the squared differences between the logarithms of power levels in corresponding channels (i.e. the square of the **Euclidean distance** in the multi-dimensional space). A graphical representation of the Euclidean distance between frames for the words used in Figure 8.1 is shown in Figure 8.2.

There are many other spectrally based representations of the signal that are more effective than the simple filter bank, and some of these will be described in Chapter 10. The filter-bank method, however, is sufficient to illustrate the pattern-matching principles explained in this chapter.

8.4 END-POINT DETECTION FOR ISOLATED WORDS

The pattern comparison methods described above assume that the beginning and end points of words can be found. In the case of words spoken in isolation in a quiet environment it is possible to use some simple level threshold to determine start and end points. There are, however, problems with this approach when words start or end with a very weak sound, such as [f]. In such cases the distinction in level between the background noise and the start or end of the word may be slight, and so the end points will be very unreliably defined. Even when a word begins and ends in a strong vowel, it is common for speakers to precede the word with slight noises caused by opening the lips, and to follow the word by quite noisy exhalation. If these spurious noises are to be excluded the level threshold will certainly have to be set high enough to also exclude weak unvoiced fricatives. Some improvement in separation of speech from background noise can be obtained if the spectral properties of the noise are also taken into account. However, there is no reliable way of determining whether low-level sounds that might immediately precede or follow a word should be regarded as an essential part of that word without simultaneously determining the identity of the word.

Of course, even when a successful level threshold criterion has been found, it is necessary to take account of the fact that some words can have a period of silence within them. Any words (such as "containing" and "stop") containing unvoiced stop consonants at some point other than the beginning belong to this category. The level threshold can still be used in such cases, provided the end-of-word decision is delayed by the length of the longest possible stop gap, to make sure that the word has really finished. When isolated words with a final unvoiced stop consonant are used in pattern matching, a more serious problem, particularly for English, is that the stop burst is sometimes, but not always, omitted by the speaker. Even when the end points are correctly determined, the patterns being compared for words which are nominally the same will then often be inherently different.

Although approximate end points can be found for most words, it is apparent from the above comments that they are often not reliable.

Ex.1019 / Page 17 of 102
AMAZON.COM, INC.

8.5 ALLOWING FOR TIMESCALE VARIATIONS

Up to now we have assumed that any words to be compared will be of the same length, and that corresponding times in separate utterances of a word will represent the same phonetic features. In practice speakers vary their speed of speaking, and often do so non-uniformly so that equivalent words of the same total length may differ in the middle. This timescale uncertainty is made worse by the unreliability of end-point detection. It would not be unusual for two patterns of apparently very different length to have the underlying utterances spoken at the same speed, and merely to have a final fricative cut short by the end-point detection algorithm in one case as a result of a slight difference in level.

Some early implementations of isolated-word recognizers tried to compensate for the timescale variation by a uniform time normalization to ensure that all patterns being matched were of the same length. This process is a great improvement over methods such as truncating the longer pattern when it is being compared with a shorter one, but the performance of such machines was undoubtedly limited by differences in timescale. In the 1960s, however, a technique was developed which is capable of matching one word on to another in a way which applies the optimum non-linear timescale distortion to achieve the best match at all points. The mathematical technique used is known as **dynamic programming** (DP), and when applied to simple word matching the process is often referred to as **dynamic time warping** (**DTW**). DP in some form is now almost universally used in speech recognizers.

8.6 DYNAMIC PROGRAMMING FOR TIME ALIGNMENT

Assume that an incoming speech pattern and a template pattern are to be compared, having n and N frames respectively. Some distance metric can be used to calculate the distance, $d(i,j)$, between frame i of the incoming speech and frame j of the template. To illustrate the principle, in Figure 8.3 the two sets of feature vectors of the words have been represented by letters of the word "pattern". Differences in timescale have been indicated by repeating or omitting letters of the word, and the fact that feature vectors will not be identical, even for corresponding points of equivalent words, is indicated by using different type styles for the letters. It is, of course, assumed in this explanation that all styles of the letter "a" will yield a lower value of distance between them than, say, the distance between an "a" and any example of the letter "p". To find the total difference between the two patterns, one requires to find the sum of all the distances between the individual pairs of frames along whichever path between the bottom-left and top-right corners in Figure 8.3 that gives the smallest distance. This definition will ensure that corresponding frames of similar words are correctly aligned.

One way of calculating this total distance is to consider all possible paths, and add the values of $d(i,j)$ along each one. The distance measure between the patterns is then taken to be the lowest value obtained for the cumulative distance. Although this method is bound to give the correct answer, the number of valid paths becomes so large that the computation is impossible for any practical speech recognition machine. Dynamic programming is a mathematical technique which guarantees to

Ex.1019 / Page 18 of 102
AMAZON.COM, INC.

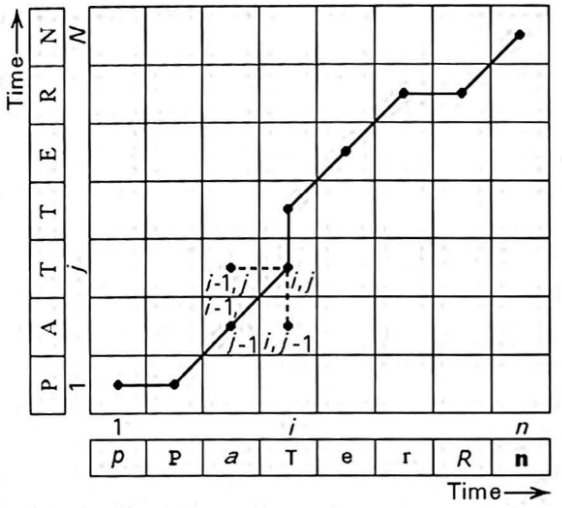


Figure 8.3 Illustration of a time-alignment path between two words that differ in their timescale. Any point i,j can have three predecessors as shown.

find the cumulative distance along the optimum path without having to calculate the cumulative distance along all possible paths.

Let us assume that valid paths obey certain common-sense constraints, such that portions of words do not match when mutually reversed in time (i.e. the path on Figure 8.3 always goes forward with a non-negative slope). Although skipping single frames could be reasonable in some circumstances, it simplifies the explanation if, for the present, we also assume that we can never omit from the comparison process any frame from either pattern. In Figure 8.3, consider a point i,j somewhere in the middle of both words. If this point is on the optimum path, then the constraints of the path necessitate that the immediately preceding point on the path is $i-1,j$ or $i-1,j-1$ or $i,j-1$. These three points are associated with a horizontal, diagonal or vertical path step respectively. Let $D(i,j)$ be the cumulative distance along the optimum path from the beginning of the word to point i,j, thus:

$$D(i,j) = \sum_{\substack{x,y=1,1 \\ \text{along the} \\ \text{best path}}}^{i,j} d(x,y) \ . \tag{8.1}$$

As there are only the three possibilities for the point before point i,j it follows that

$$D(i,j) = \min\left[D(i-1,j), D(i-1,j-1), D(i,j-1)\right] + d(i,j) \ . \tag{8.2}$$

The best way to get to point i,j is thus to get to one of the immediately preceding points *by the best way*, and then take the appropriate step to i,j. The value of $D(1,1)$ must be equal to $d(1,1)$ as this point is the beginning of all possible paths. To reach points along the bottom and the left-hand side of Figure 8.3 there is only one possible direction (horizontal or vertical, respectively). Therefore, starting with the value of $D(1,1)$, values of $D(i,1)$ or values of $D(1,j)$ can be calculated in turn for increasing values of i or j. Let us assume that we calculate the vertical column, $D(1,j)$, using a reduced form of Equation (8.2) that does not have to

Ex.1019 / Page 19 of 102
AMAZON.COM, INC.

consider values of $D(i-1,j)$ or $D(i-1,j-1)$. (As the scheme is symmetrical we could equally well have chosen the horizontal direction instead.) When the first column values for $D(1,j)$ are known, Equation (8.2) can be applied successively to calculate $D(i,j)$ for columns 2 to n. The value obtained for $D(n, N)$ is the score for the best way of matching the two words. For simple speech recognition applications, just the final score is required, and so the only working memory needed during the calculation is a one-dimensional array for holding a column (or row) of $D(i,j)$ values. However, there will then be no record at the end of what the optimum path was, and if this information is required for any purpose it is also necessary to store a two-dimensional array of back-pointers, to indicate which direction was chosen at each stage. It is not possible to know until the end has been reached whether any particular point will lie on the optimum path, and this information can only be found by tracing back from the end.

8.7 REFINEMENTS TO ISOLATED-WORD DP MATCHING

The DP algorithm represented by Equation (8.2) is intended to deal with variations of timescale between two otherwise similar words. However, if two examples of a word have the same length but one is spoken faster at the beginning and slower at the end, there will be more horizontal and vertical steps in the optimum path and fewer diagonals. As a result there will be a greater number of values of $d(i, j)$ in the final score for words with timescale differences than when the timescales are the same. Although it may be justified to have some penalty for timescale distortion, on the grounds that an utterance with a very different timescale is more likely to be the wrong word, it is better to choose values of such penalties explicitly than to have them as an incidental consequence of the algorithm. Making the number of contributions of $d(i,j)$ to $D(n, N)$ independent of the path can be achieved by modifying Equation (8.2) to add twice the value of $d(i,j)$ when the path is diagonal. One can then add an explicit penalty to the right-hand side of Equation (8.2) when the step is either vertical or horizontal. Equation (8.2) thus changes to:

$$D(i,j) = \min\big[D(i-1,j) + d(i,j) + hdp,\; D(i-1,j-1) + 2d(i,j),\; D(i,j-1) + d(i,j) + vdp\big]. \quad (8.3)$$

Suitable values for the horizontal and vertical distortion penalties, *hdp* and *vdp*, would probably have to be found by experiment in association with the chosen distance metric. It is, however, obvious that, all other things being equal, paths with appreciable timescale distortion should be given a worse score than diagonal paths, and so the values of the penalties should certainly not be zero.

Even in Equation (8.3) the number of contributions to a cumulative distance will depend on the lengths of both the example and the template, and so there will be a tendency for total distances to be smaller with short templates and larger with long templates. The final best-match decision will as a result favour short words. This bias can be avoided by dividing the total distance by the template length.

The algorithm described above is inherently symmetrical, and so makes no distinction between the word in the store of templates and the new word to be

Ex.1019 / Page 20 of 102
AMAZON.COM, INC.

identified. DP is, in fact, a much more general technique that can be applied to a wide range of applications, and which has been popularized especially by the work of Bellman (1957). The number of choices at each stage is not restricted to three, as in the example given in Figure 8.3. Nor is it necessary in speech recognition applications to assume that the best path should include all frames of both patterns. If the properties of the speech only change slowly compared with the frame interval, it is permissible to skip occasional frames, so achieving timescale compression of the pattern. A particularly useful alternative version of the algorithm is asymmetrical, in that vertical paths are not permitted. The steps have a slope of zero (horizontal), one (diagonal), or two (which skips one frame in the template). Each input frame then makes just one contribution to the total distance, so it is not appropriate to double the distance contribution for diagonal paths. Many other variants of the algorithm have been proposed, including one that allows average slopes of 0.5, 1 and 2, in which the 0.5 is achieved by preventing a horizontal step if the previous step was horizontal. Provided the details of the formula are sensibly chosen, all of these algorithms can work well. In a practical implementation computational convenience may be the reason for choosing one in preference to another.

8.8 SCORE PRUNING

Although DP algorithms provide a great computational saving compared with exhaustive search of all possible paths, the remaining computation can be substantial, particularly if each incoming word has to be compared with a large number of candidates for matching. Any saving in computation that does not affect the accuracy of the recognition result is therefore desirable. One possible computational saving is to exploit the fact that, in the calculations for any column in Figure 8.3, it is very unlikely that the best path for a correctly matching word will pass through any points for which the cumulative distance, $D(i, j)$, is much in excess of the lowest value in that column. The saving can be achieved by not allowing paths from relatively badly scoring points to propagate further. (This process is sometimes known as **pruning** because the growing paths are like branches of a tree.) There will then only be a small subset of possible paths considered, usually lying on either side of the best path. If this economy is applied it can no longer be guaranteed that the DP algorithm will find the best-scoring path. However, with a value of score-pruning threshold that reduces the average amount of computation by a factor of 5–10 the right path will almost always be obtained if the words are fairly similar. The only circumstances where this amount of pruning is likely to prevent the optimum path from being obtained will be if the words are actually different, when the resultant over-estimate of total distance would not cause any error in recognition.

Figures 8.4(a), 8.5 and 8.6 show DP paths using the symmetrical algorithm for the words illustrated in Figures 8.1 and 8.2. Figure 8.4(b) illustrates the asymmetrical algorithm for comparison, with slopes of 0, 1 and 2. In Figure 8.4 there is no time-distortion penalty, and Figure 8.5 with a small distortion penalty shows a much more plausible matching of the two timescales. The score pruning used in these figures illustrates the fact that there are low differences in cumulative

Ex.1019 / Page 21 of 102
AMAZON.COM, INC.

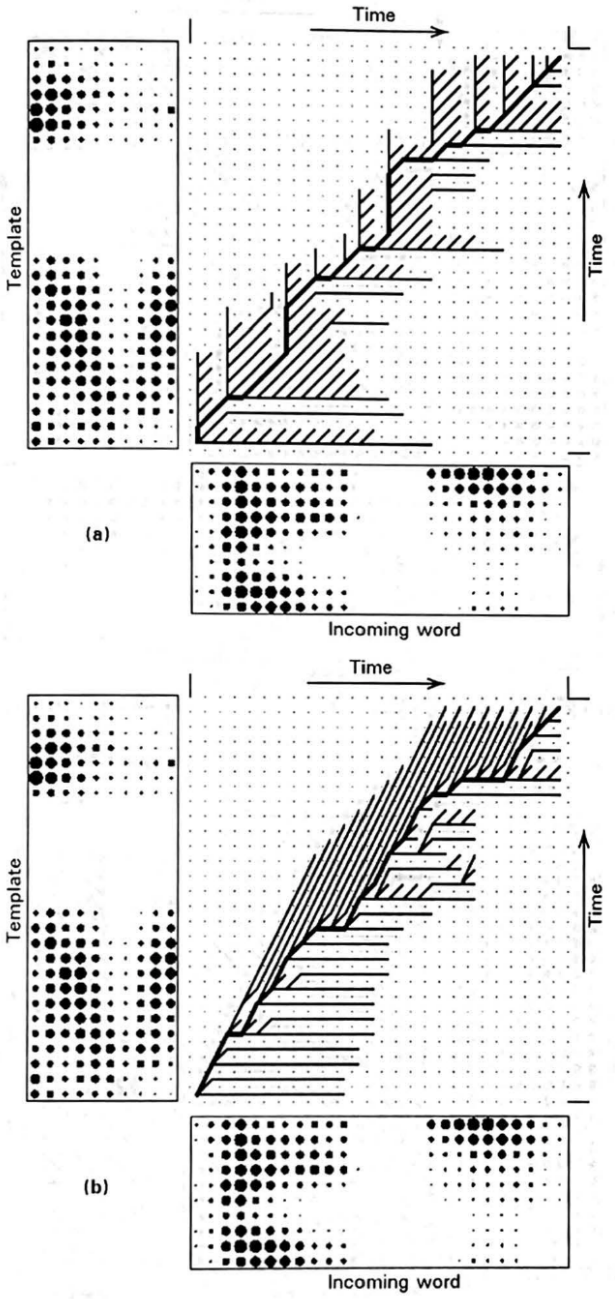


Figure 8.4 **(a)** DP alignment between two examples of the word "eight", with no timescale distortion penalty but with score pruning. The optimum path, obtained by tracing back from the top right-hand corner, is shown by the thick line. **(b)** Match between the same words as in (a), but using an asymmetric algorithm with slopes of 0,1 and 2.

distance only along a narrow band around the optimum path. When time alignment is attempted between dissimilar words, as in Figure 8.6, a very irregular path is obtained, with a poor score. Score pruning was not used in this illustration, because any path to the end of the word would then have been seriously sub-optimal.

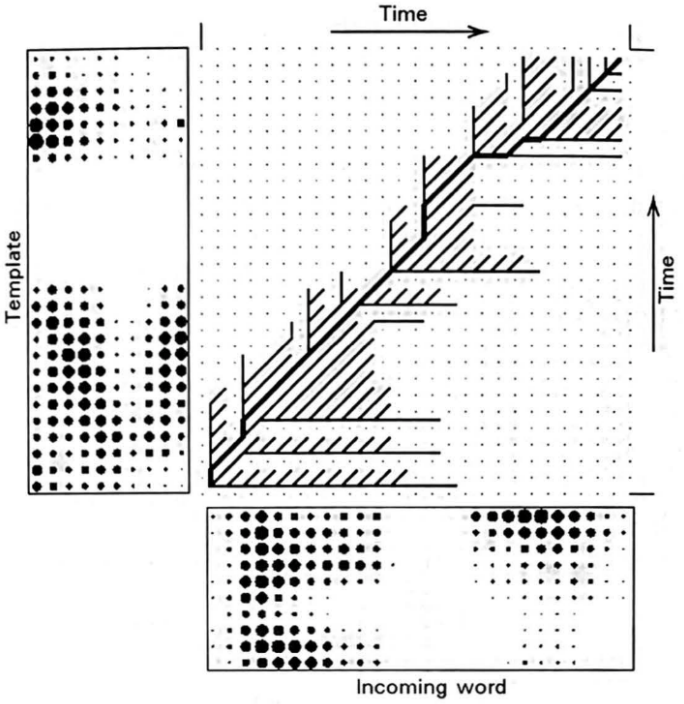


Figure 8.5 As for Figure 8.4(a), but with a small timescale distortion penalty.

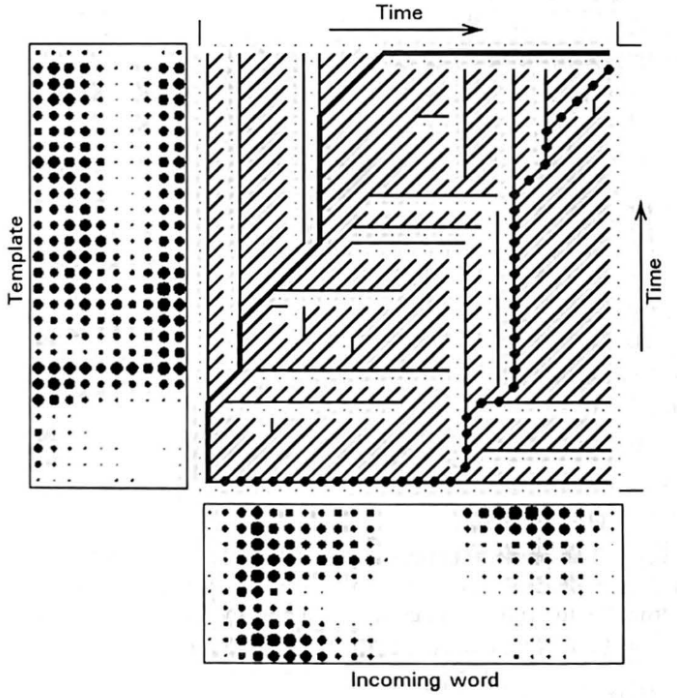


Figure 8.6 The result of trying to align two dissimilar words ("three" and "eight") within the same DP algorithm as was used for Figure 8.5. The score pruning was removed from this illustration, because any path to the end of the word would then have been seriously sub-optimal. It can be seen that if the last frame had been removed from the template, the path would have been completely different, as marked by blobs.

8.9 ALLOWING FOR END-POINT ERRORS

If an attempt is made to match two intrinsically similar words when one has its specified end point significantly in error, the best-matching path ought to align all the frames of the two words that really do correspond. Such a path implies that the extra frames of the longer word will all be lumped together at one end, as illustrated in Figure 8.7. As this extreme timescale compression is not a result of a genuine difference between the words, it may be better not to have any timescale distortion penalty for frames at the ends of the patterns, and in some versions of the algorithm it may be desirable not to include the values of $d(i, j)$ for the very distorted ends of the path. If the chosen DP algorithm disallows either horizontal steps or vertical steps, correct matching of words with serious end-point errors will not be possible, and so it is probably better to remove the path slope constraints for the end frames.

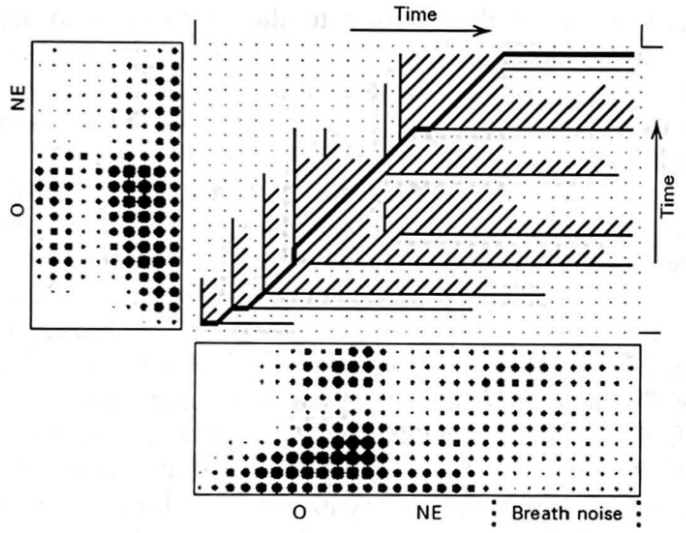


Figure 8.7 An example of the word "one" followed by breath noise, being aligned with a "one" template. A timescale distortion penalty was used except for the beginning and end frames.

8.10 DYNAMIC PROGRAMMING FOR CONNECTED WORDS

Up to now we have assumed that the words to be matched have been spoken in isolation, and that their beginnings and ends have therefore already been identified (although perhaps with difficulty). When words are spoken in a normal connected fashion, recognition is much more difficult because it is generally not possible to determine where one word ends and the next one starts independently of identifying what the words are. For example, in the sequence "six teenagers" it would be difficult to be sure that the first word was "six" rather than "sixteen" until the last syllable of the phrase had been spoken, and "sixty" might also have been possible before the [n] occurred. In some cases, such as the "grade A" example given in Chapter 1, a genuine ambiguity may remain, but for most tasks any ambiguities are resolved when at most two or three syllables have followed a word boundary.

There is another problem with connected speech as a result of co-articulation between adjacent words. It is not possible even to claim the existence of a clear

Ex.1019 / Page 24 of 102
AMAZON.COM, INC.

point where one word stops and the next one starts. However, it is mainly the ends of words that are affected and, apart from a likely speeding up of the timescale, words in a carefully spoken connected sequence do not normally differ greatly from their isolated counterparts except near the ends. In matching connected sequences of words for which separate templates are already available one might thus define the best-matching word sequence to be given by the sequence of templates which, when joined end to end, offers the best match to the input. It is of course assumed that the optimum time alignment is used for the sequence, as with DP for isolated words. Although this model of connected speech totally ignores co-articulation, it has been successfully used in many connected-word speech recognizers.

As with the isolated-word time-alignment process, there seems to be a potentially explosive increase in computation, as every frame must be considered as a possible boundary between words. When each frame is considered as an end point for one word, all other permitted words in the vocabulary have to be considered as possible starters. Once again the solution to the problem is to apply dynamic programming, but in this case the algorithm is applied to word sequences as well as to frame sequences within words. A few algorithms have been developed to extend the isolated-word DP method to work economically across word boundaries. One of the most straightforward and widely used is described below.

In Figure 8.8 consider a point that represents a match between frame i of a multi-word input utterance and frame j of template number k. Let the cumulative distance from the beginning of the utterance along the best-matching sequence of complete templates followed by the first j frames of template k be $D(i, j, k)$. The best path through template k can be found by exactly the same process as for isolated-word recognition. However, in contrast to the isolated-word case, it is not known where on the input utterance the match with template k should finish, and for every input frame any valid path that reaches the end of template k could join to the beginning of the path through another template, representing the next word. Thus, for each input frame i, it is necessary to consider all templates that may have just ended in order to find which one has the lowest cumulative score so far. This score is then used in the cumulative distance at the start of any new template, m:

$$D(i,1,m) = \min_{\text{over } k}\left[D(i-1, L(k), k)\right] + d(i,1,m)\,, \tag{8.4}$$

where $L(k)$ is the length of template k. The use of $i-1$ in Equation (8.4) implies that moving from the last frame of one template to the first frame of another always involves advancing one frame on the input (i.e. in effect only allowing diagonal paths between templates). This restriction is necessary, because the scores for the ends of all other templates may not yet be available for input frame i when the path decision has to be made. A horizontal path from within template m could have been included in Equation (8.4), but has been omitted merely to simplify the explanation. A timescale distortion penalty has not been included for the same reason.

In the same way as for isolated words, the process can be started off at the beginning of an utterance because all values of $D(0, L(k), k)$ will be zero. At the end of an utterance the template that gives the lowest cumulative distance is assumed to represent the final word of the sequence, but its identity gives no indication of the templates that preceded it. These can only be determined by storing pointers to the preceding templates of each path as it evolves, and then tracing back when the final point is reached. It is also possible to recover the positions in the input sequence

Ex.1019 / Page 25 of 102
AMAZON.COM, INC.

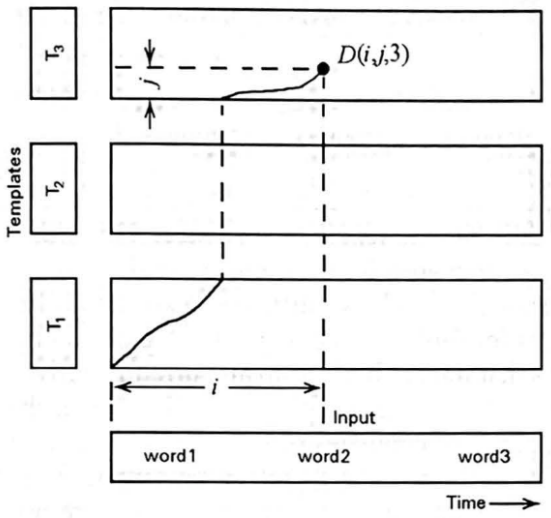


Figure 8.8 Diagram indicating the best-matching path from the beginning of an utterance to the j^{th} frame of template T_3 and the i^{th} frame of the input. In the example shown i is in the middle of the second word of the input, so the best path includes one complete template (T_1) and a part of T_3. The cumulative distance at this point is denoted by $D(i, j, 3)$, or in general by $D(i, j, k)$ for the k^{th} template.

where the templates of the matching sequence start and finish, so segmenting the utterance into separate words. Thus we solve the segmentation problem by delaying the decisions until we have seen the whole utterance and decided on the words.

The process as described so far assumes that any utterance can be modelled completely by a sequence of word templates. In practice a speaker may pause between words, so giving a period of silence (or background noise) in the middle of an utterance. The same algorithm can still be used for this situation by also storing a template for a short period of silence, and allowing this **silence template** to be included between appropriate pairs of valid words. If the silence template is also allowed to be chosen at the start or end of the sequence, the problem of end-point detection is greatly eased. It is only necessary to choose a threshold that will never be exceeded by background noise, and after the utterance has been detected, to extend it by several frames at each end to be sure that any low-intensity parts of the words are not omitted. Any additional frames before or after the utterance should then be well modelled by a sequence of one or more silence templates.

When a sequence of words is being spoken, unintentional extraneous noises (such as grunts, coughs and lip smacks) will also often be included between words. In an isolated-word recognizer these noises will not match well to any of the templates, and can be rejected on this basis. In a connected-word algorithm there is no provision for not matching any part of the sequence. However, the rejection of these unintentional insertions can be arranged by having a special template, often called a **wildcard template**, that bypasses the usual distance calculation and is deemed to match with any frame of the input to give a standard value of distance. This value is chosen to be greater than would be expected for corresponding frames of equivalent words, but less than should occur when trying to match quite different sounds. The wildcard will then provide the best score when attempting to match spurious sounds and words not in the stored template vocabulary, but should not normally be chosen in preference to any of the well-matched words in the input.

Ex.1019 / Page 26 of 102
AMAZON.COM, INC.

8.11 CONTINUOUS SPEECH RECOGNITION

In the connected-word algorithm just described, start and finish points of the input utterance must at least be approximately determined. However it is not generally necessary to wait until the end of an utterance before identifying the early words. Even before the end, one can trace back along all current paths through the tree that represents the candidates for the template sequence. This tree will always involve additional branching as time goes forward, but the ends of many of the 'twigs' will not represent a low enough cumulative distance to successfully compete with other twigs as starting points for further branching, and so paths along these twigs will be abandoned. It follows that tracing back from all currently active twigs will normally involve coalescence of all paths into a single 'trunk', which therefore represents a uniquely defined sequence of templates (see Figure 8.9). The results up to the first point of splitting of active paths can therefore be output from the machine, after which the back-pointers identifying that part of the path are no longer needed, nor are those representing abandoned paths. The memory used for storing them can therefore be released for re-use with new parts of the input signal.

The recognizer described above can evidently operate continuously, with a single pass through the input data, outputting its results always a few templates behind the current best match. Silence templates are used to match the signal when the speaker pauses, and wildcards are used for extraneous noises or inadmissible words. The time lag for output is determined entirely by the need to resolve ambiguity. When two alternative sequences of connected words both match the input well, but with different boundary points (e.g. "grey day" and "grade A") it is necessary to reach the end of the ambiguous sequence before a decision can be reached on any part of it. (In the example just given, the decision might even then

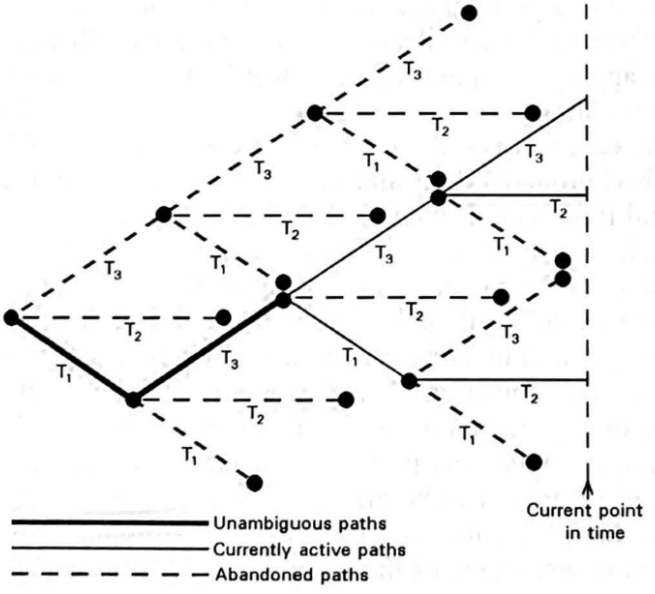


Figure 8.9 Trace-back through a word decision tree to identify unambiguous paths for a three-word vocabulary continuous recognizer. Paths are abandoned when the cumulative distances of all routes to the ends of the corresponding templates are greater than for paths to the ends of different template sequences at the same points in the input. Template sequences still being considered are T_1–T_3–T_3–T_3, T_1–T_3–T_3–T_2 and T_1–T_3–T_1–T_2. Thus T_2 is being scored separately for two preceding sequences.

Ex.1019 / Page 27 of 102
AMAZON.COM, INC.

be wrong because of inherent ambiguity in the acoustic signal.) On the other hand, if the input matches very badly to all except one of the permitted words, all paths not including that word will be abandoned as soon as the word has finished. In fact, if score pruning is used to cause poor paths to be abandoned early, the path in such a case may be uniquely determined even at a matching point within the word. There is plenty of evidence that human listeners also often decide on the identity of a long word before it is complete if its beginning is sufficiently distinctive.

8.12 SYNTACTIC CONSTRAINTS

The rules of grammar often prevent certain sequences of words from occurring in human language, and these rules apply to particular syntactic classes, such as nouns, verbs, etc. In the more artificial circumstances in which speech recognizers are often used, the tasks can sometimes be arranged to apply much more severe constraints on which words are permitted to follow each other. Although applying such constraints requires more care in designing the application of the recognizer, it usually offers a substantial gain in recognition accuracy because there are then fewer potentially confusable words to be compared. The reduction in the number of templates that need to be matched at any point also leads to a computational saving.

8.13 TRAINING A WHOLE-WORD RECOGNIZER

In all the algorithms described in this chapter it is assumed that suitable templates for the words of the vocabulary are available in the machine. Usually the templates are made from speech of the intended user, and thus a **training session** is needed for enrolment of each new user, who is required to speak examples of all the vocabulary words. If the same user regularly uses the machine, the templates can be stored in some back-up memory and re-loaded prior to each use of the system. For isolated-word recognizers the only technical problem with training is end-point detection. If the templates are stored with incorrect end points the error will affect recognition of every subsequent occurrence of the faulty word. Some systems have tried to ensure more reliable templates by time aligning a few examples of each word and averaging the measurements in corresponding frames. This technique gives some protection against occasional end-point errors, because such words would then give a poor match in this alignment process and so could be rejected.

If a connected-word recognition algorithm is available, each template can be segmented from the surrounding silence by means of a special training syntax that only allows silence and wildcard templates. The new template candidate will obviously not match the silence, so it will be allocated to the wildcard. The boundaries of the wildcard match can then be taken as end points of the template.

In acquiring templates for connected-word recognition, more realistic training examples can be obtained if connected words are used for the training. Again the recognition algorithm can be used to determine the template end points, but the syntax would specify the preceding and following words as existing templates, with just the new word to be captured represented by a wildcard between them. Provided the surrounding words can be chosen to give clear acoustic boundaries where they

Ex.1019 / Page 28 of 102
AMAZON.COM, INC.

join to the new word, the segmentation will then be fairly accurate. This process is often called **embedded training**. More powerful embedded training procedures for use with statistical recognizers are discussed in Chapters 9 and 11.

CHAPTER 8 SUMMARY

- Most early successful speech recognition machines worked by pattern matching on whole words. Acoustic analysis, for example by a bank of band-pass filters, describes the speech as a sequence of feature vectors, which can be compared with stored templates for all the words in the vocabulary using a suitable distance metric. Matching is improved if speech level is coded logarithmically and level variations are normalized.
- Two major problems in isolated-word recognition are end-point detection and timescale variation. The timescale problem can be overcome by dynamic programming (DP) to find the best way to align the timescales of the incoming word and each template (known as dynamic time warping). Performance is improved by using penalties for timescale distortion. Score pruning, which abandons alignment paths that are scoring badly, can save a lot of computation.
- DP can be extended to deal with sequences of connected words, which has the added advantage of solving the end-point detection problem. DP can also operate continuously, outputting words a second or two after they have been spoken. A wildcard template can be provided to cope with extraneous noises and words that are not in the vocabulary.
- A syntax is often provided to prevent illegal sequences of words from being recognized. This method increases accuracy and reduces the computation.

CHAPTER 8 EXERCISES

E8.1 Give examples of factors which cause acoustic differences between utterances of the same word. Why does simple pattern matching work reasonably well in spite of this variability?

E8.2 What factors influence the choice of bandwidth for filter-bank analysis?

E8.3 What are the reasons in favour of logarithmic representation of power in filter-bank analysis? What difficulties can arise due to the logarithmic scale?

E8.4 Explain the principles behind dynamic time warping, with a simple diagram.

E8.5 Describe the special precautions which are necessary when using the symmetrical DTW algorithm for isolated-word recognition.

E8.6 How can a DTW isolated-word recognizer be made more tolerant of end-point errors?

E8.7 How can a connected-word recognizer be used to segment a speech signal into individual words?

E8.8 What extra processes are needed to turn a connected-word recognizer into a continuous recognizer?

E8.9 Describe a training technique suitable for connected-word recognizers.

Ex.1019 / Page 29 of 102
AMAZON.COM, INC.

CHAPTER 9

Introduction to Stochastic Modelling

9.1 FEATURE VARIABILITY IN PATTERN MATCHING

The recognition methods described in the previous chapter exploit the fact that repeated utterances of the same word normally have more similar acoustic patterns than utterances of different words. However, it is to be expected that some parts of a pattern may vary more from occurrence to occurrence than do other parts. In the case of connected words, the ends of the template representing each word are likely to have a very variable degree of match, depending on the amount that the input pattern is modified by co-articulation with adjacent words. There is also no reason to assume that the individual features of a feature vector representing a particular phonetic event are of equal consistency. In fact, it may well occur that the value of a feature could be quite critical at a particular position in one word, while being very variable and therefore not significant in some part of a different word.

Timescale variability has already been discussed in Chapter 8. It must always be desirable to have some penalty for timescale distortion, as durations of speech sounds are not normally wildly different between different occurrences of the same word. However, there is no reason to assume that the time distortion penalty should be constant for all parts of all words. For example, it is known that long vowels can vary in length a lot, whereas most spectral transitions associated with consonants change in duration only comparatively slightly.

From the above discussion it can be seen that the ability of a recognizer to distinguish between words is likely to be improved if the variability of the patterns can be taken into account. We should not penalize the matching of a particular word if the parts that match badly are parts which are known to vary extensively from utterance to utterance. To use information about variability properly we need to have some way of collecting statistics which represent the variability of the word patterns, and a way of using this variability in the pattern-matching process.

The basic pattern-matching techniques using DTW as described in Chapter 8 started to be applied to ASR in the late 1960s and became popular during the 1970s. However, the application of statistical techniques to this problem was also starting to be explored during the 1970s, with early publications being made independently by Baker (1975) working at Carnegie-Mellon University (CMU) and by Jelinek (1976) from IBM. These more powerful techniques for representing variability have gradually taken over from simple pattern matching. In the period since the early publications by Baker and by Jelinek, there has been considerable research to refine the use of statistical methods for speech recognition, and some variant of these methods is now almost universally adopted in current systems.

This chapter provides an introduction to statistical methods for ASR. In order to accommodate pattern variability, these methods use a rather different way of