Trajectory modelling with limited speech data

J.A.C. Badenhorst

21022569

Thesis submitted for the degree *Doctor Philosophiae* in Electrical and Electronic Engineering at the Potchefstroom Campus of the North-West University

Promoter: Prof. M.H. Davel

May 2016

It all starts here ™

NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
®

Trajectory modelling with limited speech data

State-of-the-art automatic speech recognition (ASR) systems are built using hundreds or even thousands of hours of speech data. Even then, high recognition accuracy is achievable only by carefully constraining the recognition domain. This reliance on large speech corpora remains a major challenge when building ASR systems for resource-constrained languages.

The need for large corpora is partially due to the substantial variation observed in different spoken realisations of the same text but – significantly – co-articulation plays an important role. When building an ASR system, it is not sufficient to observe a large number of samples of each acoustic unit during training; it is necessary to observe sufficient samples appearing in similar contexts to those found in the test data.

To obtain a better understanding of co-articulation effects, we analysed the behaviour of phones in context, using trajectory models. We developed a new model that captures the feature trajectories of acoustic unit transitions directly, and developed a way of representing the characteristic changes between different units. We found it beneficial to model these characteristic changes at the spectral rather than cepstral level, by extracting features directly from the filter bank. Applying auto-regressive moving-average (ARMA) filtering to smooth spectral energies before constructing cepstral features also improved the accuracy of trajectories. We experimented with different approaches to identify transition model alignments and selected techniques that allowed us to locate the characteristic changes between units with the required accuracy.

We developed a new compact representation of speech units in context, estimating model parameters using the trajectory models. These models function at a sub-transitional level, enabling the construction of units that occur in unseen and rare contexts. Applying this technique, it was possible to create synthetic samples of triphone contexts, by first constructing diphone transitions and concatenating these to form synthetic trajectories. We found that better acoustic models (producing higher likelihoods on unseen test data) could be developed by augmenting existing data with synthetic samples. When the samples were used to augment the training data in an end-to-end ASR system, promising results were obtained. A useful side effect is that the synthetic samples provide a new mechanism to improve cluster selection for unseen or rare phones during state-tying.

Keywords: synthetic triphones, trajectory modelling, trajectory-based features, feature distributions, feature construction, data augmentation, resource-scarce acoustic modelling, corpus design

Trajek modellering met beperkte spraak data

Die mees gevorderde outomatiese spraakherkenning (SH) stelsels word ontwikkel deur van honderde of selfs duisende ure spraakdata gebruik te maak. Selfs dan is goeie herkenningsakkuraatheid slegs haalbaar deur die herkenningsgebied versigtig te beperk. Hierdie afhanklikheid van groot korpora afrigdata is spesifiek 'n uitdaging wanneer SH-stelsels vir tale met beperkte hulpbronne gebou moet word.

Vir dieselfde geskrewe teks, kan aansienlike akoestiese variasie in die gesproke vorm verwag word. Hierdie is egter nie die enigste rede waarom die behoefte vir groot hoeveelhede SH-data so hoog is nie: die rol wat koartikulasie speel is verseker ook van kardinale belang. Gedurende die bou van SH-stelsels is dit nie genoegsaam om 'n groot hoeveelheid voorbeelde van elke akoestiese eenheid waar te neem nie; dit is veral ook nodig om 'n genoegsame hoeveelheid voorbeelde in soortgelyke kontekste as in die toetsdata te sien.

Om koartikulasie-effekte beter te verstaan het ons die gedrag van fone in konteks geanaliseer deur van trajekmodelle gebruik te maak. Ons het 'n nuwe model ontwikkel wat die kenmerktrajekte van akoestiese eenheidsoorgange direk vasvang met 'n voorstelling van die kenmerkende veranderings tussen verskillende eenhede. Dit was voordelig om hierdie kenmerkende veranderings op 'n spektrale eerder as op 'n kepstrale vlak te modelleer. Die onttrek van kenmerke direk vanaf die filterrooster was nuttig hiervoor. Deur die tegniek van outomatiese regressiewe bewegende gemiddeld toe te pas vir die gladstryk van spektrale energie voordat kepstrale kenmerke gemaak word, kon die akkuraatheid van trajekte verder verbeter word. Ons het ge-eksperimenteer met verskillende metodes om oorgangsmodelbelynings te vind en het tegnieke gekies wat ons toegelaat het om die kenmerkende veranderinge tussen eenhede met 'n aanvaarbare akkuraatheid te beskryf.

Ons het 'n nuwe kompakte voorstelling van spraakeenhede binne konteks ontwikkel wat model parameters estimeer deur van trajekmodelle gebruik te maak. Hierdie modelle funksioneer op 'n sub-oorgangsvlak wat dan die bou van eenhede in ongesiene of skaars kontekste moontlik maak. Met hierdie tegniek was dit moontlik om sintetiese voorbeelde van trifoonkontekste te maak. Die eerste stap was om difoonoorgange the skep en saam te bind tot sintetiese trajekte. Ons vind dat beter akoestiese modelle (wat hoër sekerheidswaardes genereer vir ongesiene toetsdata) ontwikkel kan word deur sintetiese oorgange by die bestaande leerdata te voeg. As hierdie sintetiese data gebruik word om die leerdata vir 'n volledige SH-stelsel te vergroot word belowende resultate verkry. 'n Nuttige wen is dat sintetiese voorbeelde gebruik kan word om die groepering vir ongesiene of skaars foontoestande gedurende die toestandbindingsproses beter saam te bind.

Sleutelterme: sintetiese trifone, trajekmodellering, trajekgebaseerde kenmerke, kenmerkverspreidings, kenmerkskepping, data vermeerdering, hulpbron-beperkte akoestiese modellering, korpusontwerp

Contents

Abstract i

List of Figures viii

List of Tables xi

1 Introduction 1
- 1.1 The importance of context . . . 1
- 1.2 ASR for under-resourced languages . . . 3
- 1.3 Problem statement . . . 4
- 1.4 Modelling the trajectories of speech . . . 5
- 1.5 Research aims . . . 6
- 1.6 Chapter overview . . . 7
- 1.7 Conclusion . . . 8

2 Background 9
- 2.1 Introduction . . . 9
- 2.2 The current HMM paradigm . . . 9
  - 2.2.1 Attempts to overcome the temporal limitations of HMMs . . . 10
- 2.3 Achieving robust performance for HMM systems . . . 12
  - 2.3.1 Improved feature statistics . . . 12
    - 2.3.1.1 Normalisation and co-articulation . . . 13
    - 2.3.1.2 Noise robustness . . . 14
  - 2.3.2 Re-shaping feature statistics . . . 15
  - 2.3.3 Improved model training . . . 16
    - 2.3.3.1 Adaptation . . . 17
    - 2.3.3.2 Discriminative training . . . 19
- 2.4 Modelling contextual effects as trajectories . . . 21
  - 2.4.1 The role of frame-based features . . . 22
- 2.5 Augmenting limited training data . . . 23
  - 2.5.1 Semi-supervised training . . . 24
  - 2.5.2 Synthesising the data . . . 24
    - 2.5.2.1 Perturbed speech data . . . 25
    - 2.5.2.2 Speech synthesis . . . 26
- 2.6 Conclusion . . . 26

3 Co-articulation: an initial analysis 27

3.1 Introduction . . . 27
3.2 Terminology . . . 28
3.3 Tracking co-articulation . . . 28
3.3.1 Experimental data and segmentation . . . 29
3.3.2 Estimating a context-dependent unit reference and frame-based differences . . . 30
3.3.3 Calculating co-articulation trajectories . . . 32
3.4 Analysis of co-articulation effects . . . 36
3.4.1 Boundary tracking . . . 37
3.4.2 Effect of broad phonemic classes . . . 40
3.5 Conclusion . . . 41

4 Corpus design for trajectory modelling 42
4.1 Introduction . . . 42
4.2 Corpus construction . . . 43
4.3 Experimental data sets . . . 43
4.4 Recognition accuracy . . . 44
4.5 Conclusion . . . 45

5 Trajectory tracking model 46
5.1 Introduction . . . 46
5.2 Terminology . . . 47
5.3 Piecewise linear models . . . 47
5.3.1 Transition model . . . 48
5.3.2 Model fit . . . 50
5.4 Transition model improvement . . . 52
5.4.1 Connecting segments . . . 52
5.4.2 4-piece segments . . . 53
5.5 Approximation sufficiency of trajectories . . . 54
5.5.1 Model consistency . . . 55
5.5.1.1 Reference stable values . . . 55
5.5.1.2 Change descriptor consistency . . . 56
5.5.2 Features for transition modelling . . . 57
5.5.3 Evaluating trajectories . . . 58
5.5.4 Consistency of the change descriptor . . . 59
5.5.5 4-piece segments . . . 60
5.6 Conclusion . . . 63

6 Speech encoding: feature optimisation 64
6.1 Introduction . . . 64
6.2 Terminology . . . 65
6.3 Selecting level of analysis . . . 65
6.3.1 MFCCs as starting point . . . 66
6.3.2 Experimental data . . . 67
6.3.3 Segmentation of speech data . . . 67
6.3.4 Measures of approximation efficiency . . . 68
6.3.4.1 Variance-weighted MSE . . . 69

6.3.4.2 Correlation measure . . . 70
6.3.5 Comparing model approximation . . . 70
6.4 Feature-parameter analysis . . . 72
6.4.1 Feature construction . . . 72
6.4.2 Control measurements . . . 74
6.4.3 Granularity of spectral analysis . . . 76
6.4.4 Impact of frame rate . . . 78
6.5 Reducing the feature variance (low-pass filtering) . . . 78
6.5.1 Frame-based smoothing . . . 79
6.5.2 Phone recognition accuracy . . . 79
6.5.3 Changes to model estimation . . . 82
6.6 Recognition with trajectory-based features . . . 83
6.6.1 Conversion of trajectory-based test features . . . 84
6.7 Conclusion . . . 86

7 Trajectory model optimisation **87**
7.1 Introduction . . . 87
7.2 Terminology . . . 88
7.3 Model evaluation . . . 88
7.3.1 Approximating the training data . . . 89
7.3.1.1 Global metrics . . . 90
7.3.1.2 Diphone-specific metrics . . . 90
7.3.1.3 Transition-specific metrics . . . 91
7.3.2 Predicting the test data . . . 91
7.3.3 Experimental setup . . . 92
7.3.3.1 Experimental data . . . 93
7.3.3.2 Utterance models . . . 94
7.3.3.3 Experimental procedure . . . 94
7.4 Baseline analysis . . . 95
7.4.1 Predicting stable values for the test data . . . 96
7.4.2 Analysing phone transition error . . . 97
7.4.3 Transitions with large error . . . 99
7.5 Splitting phones . . . 103
7.5.1 The effect of splitting segments . . . 103
7.5.2 Predicting stable values for test data . . . 106
7.6 Improving pronunciations . . . 108
7.6.1 Results with the new dictionary . . . 108
7.6.2 Corpus segmentation . . . 110
7.7 Improving stable values on turning points . . . 111
7.8 Trajectory tracking with dynamic programming . . . 113
7.8.1 Tracking training data . . . 113
7.8.2 Improved evaluation of test trajectories . . . 114
7.8.3 Results . . . 114
7.9 Free phone recognition-based segmentation . . . 116
7.10 Conclusion . . . 118

8 Synthetic triphones **119**

8.1 Introduction . . . 119
8.2 Terminology . . . 120
8.3 Synthetic triphones . . . 120
8.3.1 Sub-phone statistics . . . 121
8.3.1.1 Channel trajectory segmentation . . . 121
8.3.1.2 Diphone segmentation . . . 122
8.3.1.3 Synthetic diphone prediction . . . 123
8.3.2 Component selection to create new n-phone . . . 125
8.3.3 Generating phone examples . . . 126
8.4 Likelihood evaluation . . . 127
8.4.1 Example-selected HMMs . . . 128
8.4.2 Log likelihoods for test phones . . . 129
8.4.3 Evaluating the log likelihood shift . . . 130
8.4.4 Choosing the best predictors . . . 131
8.5 Experimental setup . . . 132
8.5.1 Data sets and segmentation . . . 132
8.5.2 Feature trajectories . . . 133
8.5.3 Synthetic phones . . . 134
8.5.3.1 Label selection and overlap . . . 134
8.5.4 Constrained synthetic examples . . . 136
8.5.4.1 Diphone fallback . . . 136
8.5.4.2 Covariance modelling . . . 136
8.5.4.3 Re-sampling . . . 137
8.5.4.4 Split phones . . . 137
8.5.5 Adding synthetic examples for HMM estimation . . . 138
8.6 Analysing likelihoods . . . 139
8.6.1 Development data . . . 140
8.6.2 Test data . . . 143
8.7 Augmenting ASR systems . . . 144
8.7.1 Monophone analysis . . . 146
8.8 Discussion . . . 148
8.9 Conclusion . . . 149

9 Conclusion **151**
9.1 Introduction . . . 151
9.2 Summary of contribution . . . 151
9.3 Future work . . . 153
9.4 Conclusion . . . 155

A Metric definitions **156**
A.1 Transition model parameters . . . 156
A.2 Defined variable operations . . . 157
A.3 Statistical properties of parameters . . . 158
A.4 Model-feature approximation metrics . . . 159

B Algorithms used **162**

B.1 Estimating free trajectory models 162
B.1.1 Model alignment 162
B.1.2 Connecting segments 169
B.2 Predicting trajectories 170
B.2.1 Reference stable values 170
B.2.2 Predicted test features 172
B.3 Improving the stable values on turning points 173

References **177**

List of Figures

3.1 *Gradual trajectories (bottom) and MFCC frames (top) revealing strong co-articulation for the vowel-vowel phone transition.* 33
3.2 *Steep trajectory slopes (bottom) and changing MFCC values (top) revealing the definite transition of the vowel-fricative class near the ASR boundary and co-articulation effects flowing well into both phones.* 34
3.3 *Abruptly changing trajectories (bottom) and MFCC values (top) of the vowel-stop class showing little co-articulation effects (affecting only four frames).* 35
3.4 *Low separability and strong co-articulation effects yield similar MFCC values (top) and trajectories (bottom) for the nasal-nasal class.* 36

5.1 *Depiction of the transition model.* 49
5.2 *Characteristic representation for a single transition.* 50
5.3 *Piecewise linear model fit of the first four cepstra of the diphone transition /@-n/ using 3-piece transition models.* 51
5.4 *Depiction of an utterance model connecting segments by forcing stable value overlap.* 52
5.5 *Piecewise linear model fit of the first four cepstra of the diphone transition /@-n/ using 4-piece utterance models.* 53
5.6 *Depiction of 4-piece utterance model segments.* 54
5.7 *Comparing consistency $\hat{\sigma}_{trans}$ of change descriptor position on a per-cepstrum basis for free alignments. It is clear that most cepstral transitions have larger standard deviations using 3-piece models, given the depicted histograms (top and right sides of the figure) of the data.* 61
5.8 *Comparing mean duration $\hat{\mu}(p_{cs}, \omega)$ of the change descriptors on a per-cepstrum basis for free alignments. Histograms (top and right sides of the figure) clearly show the longer mean duration of change descriptors for 4-piece models.* 62

6.1 *Comparing the similarity of correlation and MSE measures regarding estimated model error (Table 6.2) across both spectral and cepstral features indicate unaffected MSE measurement for feature channel distribution shape.* 71
6.2 *Feature construction procedure optionally including trajectory modelling and frame-based filtering to create specialised MFCCs.* 73
6.3 *Phone recognition stability of linear trajectory-based systems show that higher order ARMA filtering is a viable strategy to ensure good accuracy.* . . 82

7.1 *Master train data: model fit of free trajectories for diphone transitions using the $WMSE_{diphone}$ metric, ranking these error values to reveal the number of more problematic transitions.* 97

7.2 *Master train data: model fit of free trajectories for diphone transitions using the $r_{diphone}$ metric, ranking these error values to reveal the number of more problematic transitions.* 98
7.3 *Master test data: model fit of diphone transitions using the $r_{diphone}$ metric.* 98
7.4 *Master test data: model fit of diphone transitions using the $r_{diphone}$ metric and using train set transition ordering.* 99
7.5 *Ten transitions (randomly selected from the master test set) of the 12th feature channel for the diphone transition /t=u@/, aligned with regard to the ASR boundary (black vertical line).* 102
7.6 *Analysis of the training data before splitting phones, for each broad transitional phone class d and feature channel c with the $WMSE_{trans}$ measure. Both broad transitional classes "to" (bottom) or "from" (top) of phones belonging to a particular broad phone transitional class are examined. Phones belonging to the diphthong class resulted in far larger error than other phone transitional classes.* 104
7.7 *Analysis of the training data after splitting phones, for each broad transitional phone class d and feature channel c with the measure $WMSE_{trans}$. Both broad transitional classes "to" (bottom) or "from" (top) of phones belonging to a particular broad phone transitional class are examined. The different classes of phone transitions behave much more similarly across all the feature channels.* 105
7.8 *Analysis of the test data after splitting phones, for each broad transitional phone class d and feature channel c with the measure $WMSE_{trans}$. The analysis confirms the similar behaviour of phone transitions for classes different from the training data.* 106
7.9 *Analysis of the free trajectory fit of a single feature channel, showing the 3-piece model fit (standard) and the improved 3-piece model fit (stable refit), where the fit of a stable value at turning points was improved.* . . . 111

8.1 *Channel-specific re-segmentation of trajectories into diphones (green vertical lines represent new ch_{start} and ch_{end} alignment values).* 122
8.2 *New diphone segmentation (boundaries $diph_{start}$ and $diph_{end}$) for the diphone transition /b=c/ considering all channels i.* 123
8.3 *Synthetic diphone trajectories for the synthesised diphone /b=c/.* 125
8.4 *Number of examples for triphone labels in $train_{30}$, showing that only 161 triphones occur more than 15 times.* 135
8.5 *Comparing the improvement in the mean log likelihood (γ values) of untied HMMs for different numbers of mixtures. Fewer triphone classes result in improved triphone modelling for "Hybrid : 2 Mix" and "Hybrid : 4 Mix" models.* 140
8.6 *Comparing the difference in the mean log likelihood between the untied HMMs of baseline single mixture and baseline four mixtures. Not all triphones train better four-mixture models than one-mixture models.* . . . 141
8.7 *Comparing the improvement in mean log likelihood (γ values) between, the tied HMMs of baseline single mixture and baseline eight mixtures. Most, but not all, of the triphones show improved values for eight mixture HMMs.* 142

8.8 *Comparing the difference in mean log likelihood of the sparse triphone set for tied HMMs with one- and eight-mixture components per state shows improved likelihoods could be obtained for a large number of triphone classes (94%).* 142
8.9 *Comparing the improvement in mean log likelihood (γ values) of sparse triphone labels for tied HMMs on test data. Training on both the development set and $train_{30}$ data (Train + Dev) still does not outperform the previously trained synthetic-hybrid models (Train + Synth).* 143
8.10 *Comparing phone recognition results for systems trained with and without synthetic training data. Adding synthetic examples improves system performance.* 145

List of Tables

1.1 *Typical accuracies of different sentences in a TIMIT test data set.* 2

3.1 *Terminology used throughout this work.* 28
3.2 *Number of phone transitions for which phone identities can be separated using mean frame-based values and known ASR boundaries. Centre state-level phone alignments provided even higher separability (ASR centre). In general, the Euclidean distance outperformed both the correlation and the dot product.* 32
3.3 *The number of tracked trajectories that cross (usable boundaries), analysed in terms of the average distance in frames (Diff frames) from the known transition boundaries and the goodness-of-fit ($SE_{segment}$) for different orders of polynomial functions. For all measures, the shape of the second-order function introduces additional error and the third- or fourth-order functions performed much better.* 38
3.4 *Slopes, Euclidean distance to the monophone means and the standard deviation of the slopes for third-order polynomial functions at ASR diphone transition boundary. (Ranked according to the steepness of the slopes for every broad transitional phone class.) Phone transitions with steep slopes also yield good separation for the mean difference between unit estimates.* 39
3.5 *Number of correct classifications using mean frame-based values and known ASR boundaries for specific transitions (only calculated for a subset where polynomials intersected).* 40

4.1 *Number of unique diphone transition labels in the data for various selection stages of the master training data set and test data sets. After selection, 86.7% of these diphones occurred in the test data, while 87.2% of diphone transitions had three or more examples in the training data.* . . 44
4.2 *Correctness of and alternatives for confusable phone labels, clearly displaying the derounding effect in Afrikaans.* 45

5.1 *Terminology used throughout this work.* 47
5.2 *Overall $GMSE_{diphone}$ measurements for train and test data trajectories. Both mean and standard deviation (in brackets) values show the closely similar MSE values obtained with free trajectories for MSE train and MSE test and the cost of connecting segments.* 58
5.3 *Overall $GMSE_{diphone}$ measurements for predicted stable value (with different context size options) test data trajectories. The ratios (Free fit ratio) between fixed stable value and free trajectories improve, although connecting segments increase the model error.* 59

5.4 *Overall consistency $G\hat{\sigma}_{trans}$ measurement of change descriptor position on test set. Free trajectories and 4-piece models provide better change detection.* . . . 60
5.5 *Overall $GMSE_{diphone}$ measurement on test set, when applying fixed stable values and free trajectory alignments.* . . . 60
5.6 *Overall consistency $G\hat{\sigma}_{trans}$ measurement of change descriptor durations (T_{dur}) on all data for free alignments showing the most consistent mean change descriptor durations for 3-piece models with connected segments.* . . . 63

6.1 *Terminology used throughout this work.* . . . 65
6.2 *Comparing the weighted MSE and correlation measurements for trajectory models of different intermediate features in the master test data set showing higher model error for cepstral features than when using spectral features.* . . . 71
6.3 *Baseline phone recognition results for systems trained with standard MFCCs and specialised MFCCs for which a few key options are systematically activated. The bottom part of the table includes results for systems used to tune the insertion penalty value.* . . . 75
6.4 *Effect of frame rate on phone recognition accuracy when the spectral features are linear trajectory models.* . . . 77
6.5 *Effect of different numbers of filter bank channels on the phone recognition results for control and trajectory-based (linear) systems.* . . . 77
6.6 *Effect of different frame rates on phone recognition results for control and trajectory-based (Linear) systems* . . . 78
6.7 *Development set: effect of ARMA filtering on the phone recognition results of the control and trajectory-based (Linear) systems for filters with different orders (Filter).* . . . 80
6.8 *Effect of MA and ARMA filtering on the phone recognition results of control and trajectory-based (linear) systems for filters with different orders (Filter). Correlation (Cor) and MSE measures clearly indicate ARMA filters to be more effective to generate smooth features which are better approximated by linear trajectory models. Phone recognition accuracies remain near optimal for higher filter orders (4 to 8) and using semi-tied transforms.* . . . 81
6.9 *Effect of MA and ARMA filtering on model estimation detected by estimating MSE and correlation (Cor) measures for "control" features instead.* 83
6.10 *Phone recognition results showing reduced mismatch for trajectory-based systems and filtered (smoothed) test features setting insertion penalties on the development set (Dev).* . . . 83
6.11 *Master test set with optimised insertion penalties (IP): phone recognition results showing reduced mismatch for trajectory-based systems and filtered (smoothed) test features across a broad range of filter orders.* . . . 84
6.12 *Effect of converting test data to the same trajectory-based features using the phone alignments of the original recognition system, setting insertion penalties on the development set (Dev). Converting to trajectory-based front-end features provides accuracy close to the control.* . . . 84
6.13 *Master test set with optimised insertion penalties (IP): effect of converting test data to the same trajectory-based features using the phone alignments of the original recognition system.* . . . 85

6.14 *Master test set with optimised insertion penalties (IP): converting the test data to the same trajectory-based features using the phone alignments of first-pass recognition and performing a sweep of ARMA filter orders.* . . . 86
6.15 *Converting the test data to the same trajectory-based features, using the phone alignments of first-pass recognition and setting insertion penalties on the development set (Dev). A slight improvement in accuracy (90.44%) is achieved compared to the accuracy for merely smoothing the test features (89.99%) shown in Table 6.10.* . . . 86

7.1 *Terminology used throughout this work.* . . . 88
7.2 *Model fit between the free trajectories and the feature values of the master train data and the test data sets clearly shows a benefit for activating the ARMA filtering option.* . . . 96
7.3 *Mismatch between the baseline-predicted trajectory options and the actual feature values of the test data set. ARMA filtering improves the predictability of the frames of the 3-piece utterance models.* . . . 96
7.4 *Diphone transitions with high $WMSE_{diphone}$ error (at least 2σ from the $GWMSE_{diphone}$ value) for 3-piece (ARMA 6) models.* . . . 100
7.5 *Word boundary analysis of diphone transitions with a high $WMSE_{diphone}$ error showing a limited set of "unexpected" diphone transition labels formed between words (top part of the table).* . . . 100
7.6 *Number of examples for diphone labels consisting of repeated phones, showing that most of these transitions occur between words.* . . . 101
7.7 *Comparing the model fit between the free trajectories and the feature values of the master train data and test data sets show that splitting segments improve model approximation.* . . . 104
7.8 *Diphone transitions with high $WMSE_{diphone}$ error for 3-piece (ARMA 6) models after splitting the phones of the stops and diphthong classes showing fewer transitions than the previous result (Table 7.5) for a lower cut-off value.* . . . 107
7.9 *After splitting phones: the number of examples seen for diphone labels consisting of repeated phones remains fairly similar to what was detected previously (Table 7.6).* . . . 107
7.10 *Comparing the mismatch between predicted trajectories and the actual feature values of the test set for 3-piece (ARMA6) models with and without split phones. Splitting the segments produced more accurate test trajectories.* 107
7.11 *Model fit between the free trajectories and the feature values of both the master train data and test data sets, using the improved pronunciation dictionary, do not show an effect on model approximation.* . . . 109
7.12 *Model tracking of the predicted test trajectories and the test features for the improved pronunciation dictionary show that global prediction accuracy remains similar.* . . . 109
7.13 *Comparing phone recognition results for ARMA 6 systems trained by using the old and the updated pronunciation dictionaries indicates slightly higher recognition accuracy for the system trained with the new pronunciation dictionary.* . . . 109

7.14 *Model tracking of the predicted test trajectories and the test features showing the effect of corpus segmentation for trajectory modelling. Performing no filtering during segmentation, but ARMA filtering of features for trajectory estimation (scenario 3) is the better choice.* 110
7.15 *Model fit between the free trajectories and the feature values of the master train data and test data sets using the improved stable value fits show improved model approximation when applying the "RefitStable" algorithm.* 112
7.16 *Effect on accuracy of predicted test trajectories when applying the "RefitStable" algorithm during training. Almost not change is detected.* 112
7.17 *Model fit between the free trajectories and the feature values of both the master train and test data sets using the dynamic programming algorithm and improved stable value fits. As before, incorporating the "RefitStable" algorithm provides the best model approximation (DP + Stable refit).* . . . 115
7.18 *Effect on accuracy of predicted test trajectories when using the new alignment strategy for the evaluation of reference stable value predictors. A considerable reduction in model error is achieved.* 115
7.19 *Model fit between the free trajectories and the feature values of the master train data and test data sets using forced alignment to segment the train data and free phone recognition to segment the test data remains stable.* . 116
7.20 *Effect on the accuracy of predicted test trajectories when free phone recognition generates a phone sequence for test data segmentation. Forced aligning test data with a recognition-based reference (No oracle + Aligned split) restores modelling accuracy.* 117

8.1 *Terminology used throughout this work.* 120
8.2 *The number of utterances in our previous well-resourced data sets, as well as their estimated duration in minutes.* 132
8.3 *The number of triphone labels per label category.* 135
8.4 *Comparing phone recognition results for systems trained with and without synthetic training data and adjusting insertion penalties on the development data (Dev).* 144
8.5 *The effect of adding synthetic examples to training data on phone correctness, comparing the synthetic and control systems. Most monophones that show improved correctness for the development data also show improved correctness in the test data (Improved phones: Dev). For a list of seven phone labels the test data show improved phone correctness and the development set result does not (Additional improved phones: Tst).* 147
8.6 *Restricting synthetic training examples to monophone classes that improve the development set results still produced improved recognition results, but not to the extent of using all the synthetic examples chosen by the likelihood analysis.* 148

A.1 *Trajectory model parameters describing a single transition.* 156
A.2 *Frame-based trajectory model parameters (in number of frames) calculated from the values in Table A.1.* 157
A.3 *Variable operations.* 157
A.4 *Statistical properties that are used to describe and analyse parameters and parameter estimators throughout this work.* 158
A.5 *Model-feature approximation measurements.* 159

A.6 *Model-feature approximation measurements.* 160
A.7 *Global metrics to analyse the model-feature approximation of a data set.* . 161

Chapter 1

Introduction

There is general agreement that the current automatic speech recognition (ASR) technology requires large amounts of training data to achieve high accuracies in speech-recognition systems: state-of-the-art large-vocabulary systems are trained by using hundreds to thousands of hours of data. It is not clear, however, why so much data is required: is it because of the inherent variability in speakers, channel conditions, speaking styles, etc., or because of the complexity of representing cross-phone co-articulation accurately, or for some other reason? This issue is theoretically important and also crucial to the development of systems in resource-constrained environments.

This chapter introduces the rationale for studying the effects of context on ASR systems, and gives reasons for the analysis of these effects, using the trajectory tracking of speech models. The main goals of the thesis are defined (Section 1.5) and a chapter overview appears in Section 1.6.

1.1 The importance of context

The performance of typical hidden Markov model (HMM) systems on different subcorpora of the TIMIT corpus [1] gives interesting insight into the data requirement issue. In particular, we have repeatedly found that performance is substantially better on what is called speaker-independent sentences (the *sa* subset, where all training and testing speakers record the same prompts), compared with speaker-dependent sentences (the *si* subset, where different speakers record different prompts and therefore each of the sentences is recorded only once). Table 1.1 lists the phone recognition accuracies obtained for subsections of the testing data containing the indicated sentences. All the accuracies were obtained by using the same HMMs, constructed from the training set.

(Table 1.1 also contains the results for the *sx* sentences, which were read by small subsets of the speakers – these sentences clearly behave similarly to the speaker-dependent sentences.)

Since these sub-corpora are subject to the same intra- and inter-speaker sources of variability, the large difference in accuracy between the *sa* sentences and the other two sentence types suggests that context modelling (and therefore co-articulation) plays a significant role in the accuracy of speech-recognition systems and therefore also in their need for large training corpora. It is clearly not enough to see a sufficient number of phone samples; it is necessary to see enough samples in contexts sufficiently similar to those observed in the testing data.

Subset	**Gender**	**% Accuracy**
sa	male	88.78
sa	female	87.47
si	male	61.24
sx	male	61.13
sx	female	57.46
si	female	56.20
Total	-	65.28

TABLE 1.1: *Typical accuracies of different sentences in a TIMIT test data set.*

As early as the first large-vocabulary speaker-independent continuous speech recogniser, it had been recognised that contextual effects were a crucial consideration when training HMM-based acoustic models. The SPHINX system could achieve reasonable recognition accuracies, using carefully designed representations of phone modelling [2]. One problem with modelling techniques using a phone representation is that it is harder to account for longer-term effects. Longer-term pronunciation effects may be significant; evidence from the speech production process suggests the existence of underlying articulatory trajectories in speech data [3]. As a result, much research in spoken language technology is intended to incorporate the structures of human speech and language into current statistical speech recognition systems [4].

If unlimited training data were available, it would surely be more beneficial to model co-articulatory effects using whole-word units instead of phones as the basic modelling unit. In speech recognition, co-articulation effects are completely captured by the within- and cross-word contexts. It is the limited training data scenario that compels one to resort to using smaller units, such as phones and context-dependent phones, to approximate the within-word co-articulation effects for larger grammars [5]. Working with such small units entails an entirely new set of challenges. In fact, the key motivating factor for the later development of segmental models was the opportunity to exploit the acoustic features that become apparent only at the segmental level, not at the frame level. For

these models, it is important to handle the extra-segmental variability (between different examples of sub-phonemic speech segments) and the intra-segmental variability (within a single example) accurately [6].

In a more recent approach to countering the deteriorating effects of variability, speech scientists have performed what is called "detection-based ASR". This procedure uses conditional random fields (CRFs) to combine the recognition results of different ASR systems [7]. By using phonologically optimised feature sets for the phone recognition tasks, each detector can focus on various (complementary) aspects of the same speech signal. The technology for integrating segmental conditional random fields was made available in a recently released SCARF toolkit [8]. The SCARF approach allows one to integrate, in a flexible way, multiple information sources to augment the results of speech recognition. In fact, it is now feasible to combine detector output at different granularities, from frame level to phone level or up to word level [9].

1.2 ASR for under-resourced languages

Roughly counting, there are about 6 000 spoken languages in the world, of which only a limited few have developed human language technology (HLT) resources and high-quality ASR systems. Today, information technology is becoming of increasing importance in developing countries. In addition, many more languages have become of interest to HLT development because of economic and political reasons. In [10] the concept of a "computerization level" is used as a metric to describe the HLT readiness of a particular language. The authors' analysis provides a list of scored services used to evaluate the "computerization level". Developing services such as ASR and text-to-speech systems (TTS) is difficult because it requires large amounts of resources and depends on the availability of other HLT services.

Currently, the Babel project aims to develop methods to construct speech recognition systems for an increasing set of HLT languages more rapidly [11]. By improving the capability of keyword search (KWS), also called spoken term detection (STD), researchers attempt to fast-track the development of speech systems [12]. The initiative took off when the US National Institute of Standards and Technology created an STD research programme in 2006 to process archived speech data. One of the research outcomes drew attention to the fact that close relationships between KWS and state-of-the-art speech recognition performance exist for a combination of language and genre [12].

1.3 Problem statement

Building high-quality speech recognition systems for developing countries in the world can be particularly challenging, because of the limited speech data resources for under-resourced languages. Furthermore, the acoustic variability arising from different speakers, variable speaking styles and contextual effects has to be modelled correctly because it severely complicates the development of large speech corpora for new languages. Specifically, the huge effort required for data collection stems from the fact that the whole process of acquiring or developing specific texts and pronunciation dictionaries, performing careful text selection and recording usable audio samples has to be performed effectively by covering sufficient contexts.

Trajectory modelling may provide a way to leverage additional contextual information without requiring as much training data. The reason for this optimism is that the poor modelling of contextual effects contributes to the above-mentioned data hunger. Current systems based on the HMM do not model temporal (inter-frame) correlations explicitly. In practice, the context sizes of three (triphones) or five (quinphones) are often used instead. When data is limited, many of these context-dependent units will rarely or never be seen during training. In typical ASR systems, such unseen context-dependent units are modelled by clustering them with "matching" seen units, based on a combination of acoustic and linguistic analysis, which is not always an optimal solution [13]. We were interested in determining whether it would be possible to generate synthetic versions of such unseen or rare contexts from the less-specialised units observed in the training data.

First, we required a model that would link the more general units to the more specialised units. To this end, we intended to use a trajectory model that provided a compact way of representing the characteristic behaviour of transitions. Then it was possible to reconstruct models for unseen transitions from the characteristic trajectory behaviour of the less-specialised transitions. We foresaw that the current study could be restricted to triphone modelling, so we aimed to generate synthetic triphones from seen diphones. If this were possible, it should be possible to apply the same approach to larger contexts, and possibly also to synthesise additional speech data based on a small sample of data from a given speaker.

As more data from well-resourced languages becomes available, a more detailed analysis of trajectories and the variables that influence these may become possible. Such an analysis could inform the development of techniques appropriate to trajectory modelling in resource-scarce environments. For example, it might be possible to apply trajectory models trained in well-resourced languages to under-resourced languages. Though many

techniques have been developed that are applicable to trajectory modelling in well-resourced environments, the application of these models to data-scarce environments has not yet been well studied. It is therefore not yet known whether trajectory modelling with extremely limited data could result in improved acoustic models for ASR purposes, and consequently, in improved ASR results when speech data is severely constrained.

1.4 Modelling the trajectories of speech

A trajectory model provides a mechanism for capturing explicitly the slow-varying temporal changes of speech data. These changes are due to the gradual progression of one phone context to the next during the spoken utterance. Since the speech production process places constraints on how the speech signal changes at any particular point during an utterance, one phoneme cannot change instantaneously to the next. The pronunciation of a phoneme is always influenced by the previous and the next phonetic contexts.

Speech systems require many phone classes when modelling speech signals; this is largely due to co-articulation. These complex systems have to represent the many sources of variability in pronunciation accurately. Starting with a single phone, intra-phone variability occurs in different examples of the same context of a single speaker. Furthermore, the speech of a speaker contains many phone combinations. These combinations substantially increase the number of phone contexts. Also, the fact that the vocal tract lengths of speakers differ from one speaker to another creates many additional examples. Finally, it is true that pronunciation can be idiosyncratic: speakers of the same language may produce and co-articulate the same contexts differently. VTLN variation is only one of the factors influencing inter-speaker variation. The idiosyncratic category is diffuse and important (though not well modelled by current methods).

The contextual effects created by co-articulation are important to ASR and TTS systems. Therefore, the development of large-vocabulary speech recognition has long since required the use of triphone [14] or even quinphone models. In these systems, contexts are modelled implicitly within the more general statistical (HMM) framework. TTS systems that were soon developed extend this approach by means of the HMM-based speech synthesis system (HTS) [15]. HTS approaches have extended the already large phonetic structure that has to be maintained. The features modelled currently are not only spectrum-based (using mel-cepstral coefficients); the additional context-dependent features for excitation (fundamental frequencies $F0$) and their dynamic features require more acoustic classes to be modelled.

Though HMM-based speech recognition systems currently achieve acceptable performance for constrained domains, less controlled recording conditions and speaking styles have been shown to be a problem. Significant deterioration in accuracy has been measured. The extent of such decreases in performance suggests that there may still be inherent deficiencies in the current acoustic modelling paradigm [6]. As a result, the systems built on this modelling paradigm still require large amounts of training data to account for differences at the contextual level.

Trajectory modelling may provide a way to extract additional contextual information and reduce the data requirement of standard HMM systems. The component of the speech signal that carries information about contextual change is a slow-varying signal (typically below a frequency of about 60 Hz). Since the work on the temporal modulation of cepstral trajectories supports this idea [16], trajectory models may be an effective way of dealing with this kind of variation.

1.5 Research aims

In general we wanted to investigate the use of frame-based trajectory modelling techniques to alleviate some of the current requirements for extensive speech corpora when developing ASR systems. To accomplish this goal, it was necessary to know:

- What type of speech coding enables the successful representation of contextual (speech) information in a way that is suited to further trajectory-based analysis?
- Can the features of the selected speech coding be simplified so that they can be represented by linear functions (trajectories) in time?
- Can contextual information be shared by using the developed trajectory data representation to simulate additional training examples?
- Can speech recognition results be improved through trajectory-based analysis?

Our basic hypothesis is that the explicit modelling of frame-based trajectories can lead to improved acoustic modelling when data is limited, by supporting different ways in which information can be shared across context, speaker and/or language boundaries, leading to improved ASR performance in resource-scarce environments.

1.6 Chapter overview

Chapter 2 gives an overview of the literature on the various ways in which the HMM paradigm has been refined and improved, providing greater modelling accuracy. Improving and reshaping feature statistics reduce model mismatch. Model-based adaptation approaches do the same. Discriminative training and trajectory modelling share interesting synergies. All these developments form part of a steady stream of improvements made to acoustic modelling. More recently, research aims at extending the available training samples by means of data augmentation approaches. As far as we know, these approaches have not yet included the use of trajectory information.

We explain our initial analysis to investigate the effect of co-articulation in Chapter 3. The behaviour of specific phone transitions might be tracked by analysing the distance of frames from reference unit estimates. We obtained such estimates from multiple examples of the same phone transition types. Grouping multiple transitions with regard to broad phonemic classes then make it possible to show how co-articulation occurs differently in certain categories. The intricacy of the observed set of effects led to the realisation that a high-quality speech corpus would be best for experimentation from this point onward. Chapter 4 describes how the Afrikaans Trajectory Tracking corpus (ATTC) was created and what data set selections were used in subsequent experiments.

Chapter 5 explains how we established an approach to modelling phone transitions. Many of the feature trajectories of diphone segments display this definite transitional behaviour and then remain more stable near phone centres. We elected to track these characteristic changes using piecewise linear models. The feature trajectories of Mel frequency cepstral coefficients (MFCCs) are widely used to train ASR systems, but these are still not optimal to represent the transitions we modelled. Chapter 6 details how we created new trajectory-based ASR features that would be more suitable for trajectory modelling.

The simple piecewise linear approximations with which we modelled phone transitions still introduced errors, even with the new ASR features of choice. These shortcomings became more specific to the identities of the exact phone examples involved, because the acoustic quality of diphone transitions varies widely. The goal in Chapter 7 was to find and correct any remaining large errors of transitions. To this end, we made various improvements, ranging from more specialised unit segmentation to better algorithms for estimating the feature trajectories.

Since achieving good ASR accuracy relies strongly on seeing sufficient numbers of phone samples in sufficiently similar contexts to those observed in the testing data, we chose to experiment with synthetic phone transitions. ASR systems deal with under-resourced

contexts by forming (more general) clusters from examples of similar context. If the synthetic examples we constructed could allow better (more specific) clusters to form, ASR accuracy should improve. Chapter 8 defines our approach to generating synthetic triphones from diphone and even monophone trajectories. Finally, evaluating the likelihood of test data for the new ASR models we trained on the augmented training data sets confirmed the improved modelling of speech data.

1.7 Conclusion

This introduction sets the scene for the research described in this thesis. The next chapter sketches the background of acoustic modelling in speech systems and discusses shortcomings in these approaches when used in a low-resource setting.

Chapter 2

Background

2.1 Introduction

This chapter gives an overview of the acoustic modelling of speech data. In particular, we focus on training the HMM and its ability to represent speech data accurately. These models are widely used in ASR and also in TTS systems. Section 2.2 introduces the limitations of standard HMMs to model speech data. It follows that new research on building more robust HMM speech systems has extended HMMs in many ways. Section 2.3 describes and groups these techniques into three main categories:

- Improving feature statistics
- Re-shaping feature statistics
- Improved model training.

Section 2.4 explains how trajectory models (which explicitly focus more on the temporal information in training data) have further improved training. Data augmentation is a more recent development to address the limitations of training data more directly. We present a few of these approaches in Section 2.5.

2.2 The current HMM paradigm

The current HMM framework can be viewed as broadly appropriate for modelling speech patterns and is successful in accommodating the time-scale as well as short-term spectral variability. These models alone do not, however, take advantage of the constraints

inherent in the speech production process and consequently make assumptions that are inappropriate when modelling speech patterns. The state-based independence assumption is of particular interest. This assumption implies that the observation output probability is conditionally independent of all other observations, given a specific HMM state. Consequently, temporal (inter-frame) correlations are poorly modelled, as the authors state in [3]: "*The use of an independent and identically distributed (i.i.d.) stochastic process (conditioned on the HMM state sequence) as the acoustic interface model disregards many key temporal correlation properties in the acoustic signal resulting from relatively smooth motion of the articulatory structures.*"

These limitations of the HMM create similar constraints on TTS systems. Here, generating more accurate acoustic observations also requires the temporal correlation properties to be left intact. In this regard, the authors of [17] state: "*The HMM only provides a coarse approximation of the underlying process for the generation of acoustic observations, in particular, the conditional independence assumption of acoustic features and the first-order Markovian assumption for state transitions. Consequently, numerous models have been proposed that attempt to overcome the shortfalls of the HMM and provide better performance with respect to ASR and TTS.*"

2.2.1 Attempts to overcome the temporal limitations of HMMs

The simplest way of addressing the above limitations of HMMs is to add change information as additional ASR features. Exactly the same HMMs can be trained and can include dynamic features [18]. Although the authors of [17] call this technique "the most elementary effort to improve HMM modeling", including these features has a significant impact on ASR and TTS performance. The importance of this technique becomes clearer when one considers the way in which statistical parametric speech synthesis (HTS) operates. Before speech can be synthesised, HTS requires inference of the observation vectors, which exploit the explicit relationship between dynamic and static features [19]. Since the temporal characteristics are such an important consideration for TTS, research has been done that also modifies the HMM to provide the more explicit modelling of state durations. In a hidden semi-Markov model, each state can now emit a sequence of observations, instead of only a single observation per state. This process explicitly defines a variable duration for each state.

It is possible to redefine the HMM as a special case of a dynamic Bayesian network or more generally as a graphic model. This complementary representation is particularly useful for describing a variety of model extensions. Using the dynamic Bayesian network approach, researchers have attempted to induce the underlying acoustic model structure

from speech data, possibly obtaining a structure more suited to the training data [20]. Furthermore, using the new dynamic Bayesian network structure, it is possible to show how proposed model extensions modify the conditional independence assumptions of the HMM. In [21] the authors state that two approaches may be combined to extend the HMM model structure. By adding additional dependency arcs between variables, a dynamic Bayesian network can be created that describes HMMs with explicit temporal correlations between states, HMMs with vector predictors and lastly to create the buried Markov model. Adding more unobserved variables allows for dynamic Bayesian network models that are equivalent to HMMs with Gaussian mixture model (GMM) state distributions. In this way, dynamic Bayesian networks can also be created for an HMM with factor-analysed covariances.

Using full covariance matrix HMMs can model intra-frame correlation better, but building large systems this way is difficult because of the sheer size of all the parameters such systems require. A more compact way to obtain improved modelling of intra-frame correlation is to use a factor analysis based observation process [22]. These factor-analysed HMMs combines the observation process from a shared factor analysis with the standard diagonal covariance GMMs of HMM states to act as a state evolution process. Sharing information between HMM states and thus relaxing the conditional independence assumption is an alternative option. In subspace Gaussian mixture models (SGMMs) a joint structure is shared between all HMM states in an ASR system. The SGMM also uses a GMM distribution to model the characteristics of each HMM state. However, instead of specifying the parameters directly, the technique combines a vector representation of a state with a global mapping. In this way state-specific probability density functions (PDF) are still obtained. It is the global mapping from a shared S-dimensional vector space that spans across all shared states [23].

More recently a hybrid structure has been used to improve further speech recognition results of large-vocabulary speech recognition (LVSR) systems. In their work [24] combine deep neural networks (DNNs) with HMMs, so that the DNN models the observation likelihoods for each HMM state. If the DNN is better able to predict these observation likelihoods than the GMMs used with standard HMMs, this is a viable approach. Indeed, replacing the GMMs of every state with these more powerful predictors does work. The HMM structure then still models the sequential nature of the speech. Discriminative training of HMMs provides gains for the same reason. In Section 2.3.3.2, we discuss how these training approaches operate. It is significant that, trajectory modelling can also be seen as a discriminative training approach, one with an explicit temporal dependency. We now continue to provide a more in-depth discussion of the major techniques used to ensure robust ASR performance.

2.3 Achieving robust performance for HMM systems

Simply relying on the approximations and simplifying assumptions of the HMM framework when building large-vocabulary continuous speech recognition (LVCSR) systems would result in poor accuracy and oversensitivity of the system to changes in the operating environment. Obtaining high accuracy for LVCSR systems requires various system refinements. In [21] Gales provides the following list: "*feature projection, improved covariance modelling, discriminative parameter estimation, adaptation, normalisation, noise compensation and multi-pass system combination.*" Applying most of the techniques mentioned in Section 2.2.1 to modify HMMs and overcome the limitations of the modelling structure yields improvements in speech recognition systems. Clearly, this is not the only dependency for speech recognition accuracy. Supplying the optimum set of features that complements a particular modelling structure is key when the input signal originates from real speech data and is encoded as high-dimensional patterns [25].

Feature extraction transforms the signal into meaningful values so that classification can be carried out. In general, it is true that the same classifier can be used for different tasks, as long as the front-end feature extraction part is specifically fine-tuned to the task at hand. Since HMMs model the means and covariances of the speech frames assigned to every state, the distribution of speech frames is a significant factor. These feature distributions are adversely affected by noisy conditions and introduce signal mismatching. The next section describes the main approaches used to improve feature statistics, transforming the features to fit the assumptions of HMMs more closely. Section 2.3.3 describes what this approach can accomplish. Finally, improved parameter estimations are possible and we discuss these approaches in Section 2.3.3.

2.3.1 Improved feature statistics

Firstly, the mean and covariance of every segment of the speech frames matter when mapping to an HMM state to obtain acceptable recognition accuracy. Secondly, when we train the HMMs of a speech recognition system, the quality of these feature statistics also influences segmentation accuracy when associating speech frames with particular HMM states. For this reason, it is crucial to use features with optimal statistics. Mismatches between training and testing conditions may cause drastic deterioration in accuracy. Unfortunately, various real-world conditions, which include different channel characteristics and ambient background noises, distort the feature distributions of ASR features to a significant extent.

2.3.1.1 Normalisation and co-articulation

The simplest way of creating a better match is to attempt to collect data from a broad range of acoustic environments. This approach does improve the noise robustness of systems, but collecting such a huge and diverse set of data is very difficult and costly. Furthermore, training HMMs on such a data set often leads to large variances, which moreover do not provide high accuracy for any particular environment [26]. An opposite approach is to normalise the output of the feature extraction process to obtain equal segmental parameter statistics. Instead of trying to account for the feature shifts of all acoustic environments, we minimised the effect of the different environments by normalising each segment so that it would have more similar characteristics.

Researchers have reported substantial improvements in environment mismatching when using only cepstral mean normalisation. In addition, the authors of [26] claim that it is also important to normalise the feature variances to accommodate the changing conditions of real-world data. Spectral analysis shows that segmental normalisation allows spectrograms of clean and noisy utterances to look more similar than in the case of the original MFCCs. The advantages of this method include that it adapts quickly to changing conditions, requires no prior knowledge of noise statistics and requires no voice activity detection.

Cepstral mean and variance normalisation has also been employed in the model domain [27]. In this approach, the segmentation of the unnormalised features occurs first, before estimating the context-specific normalisation parameters of each HMM state. The technique could potentially provide better state-specific statistics in terms of co-articulation. The downside is that channel mismatch of the unnormalised features may adversely affect segmentation and, together with data scarcity, result in a less robust system. In principle, improving the feature statistics by taking into account the co-articulation effects while also retaining efficient data sharing among segments (to estimate robust normalisation parameters) should be best. It is trajectory modelling that provides a way to relate the effects of co-articulation and normalisation.

Co-articulation characteristics are speaker-specific. One of the primary reasons for this is that the vocal tract length differs among speakers. As a result, the formant frequencies of the spectrum shift in a linear manner among speakers. These shifts in the frequency of the components carrying most energy create an additional mismatch in ASR features. A number of approaches to vocal tract length normalisation (VTLN) have been taken to counteract this effect [28, 29], by attempting to modify the incoming audio so as to reduce differences between productions of phones. The effect of VTLN can also be approximated by using a linear transform. This last-mentioned method works well,

since it is then possible to estimate optimal transformation parameters by making only a single pass over the speech data [30, 31]. Wide coverage of different VTLN approaches is summarised in [32].

2.3.1.2 Noise robustness

Stereo-based stochastic mapping is an approach used for "recovering" corrupted speech frames. By jointly modelling the frames of noisy and clean training data sets with a GMM, the authors of [33] predict "clean" frames of test data. They use two predictors. The first is based on maximum *a posteriori* (MAP) estimation and the second on minimum mean square error estimation. Determining the resulting linear transformation of the test data from the parameters of the joint distribution works for both these techniques, but the standard stereo-based stochastic mapping approaches still do not model the dynamic feature components correctly. To improve the dynamic feature mapping characteristics, Zen, Nankaku and Tokuda [34] use the trajectory-based HMM [35] to estimate the phone-specific GMMs. Then they use these GMMs to estimate PDFs.

Chen and Bilmes [36] use auto-regression moving-average (ARMA) filtering to smooth the cepstra and thus improve ASR robustness in noisy conditions. Most of the information in speech is in the low-frequency spectral modulation. This observation is easy to explain: human speech cannot change in an arbitrary fast way because of anatomical constraints. Therefore effectively using a low-pass filter to smooth the cepstral features works well to reduce the signal mismatch of higher frequency components. The experiments Chen and Bilmes conducted on the Aurora 2.0 noisy speech database show that including mean subtraction as well as variance normalisation before the ARMA filtering step yields the best results. Furthermore, the improvements made in average error rates are comparable with far more complicated noise robustness techniques.

Improving the statistics of cepstral features such as MFCCs or perceptual linear prediction coefficients (PLPs) does work. If the noise effect can be isolated earlier during the feature construction process, however, this effect could be easier to remove [16]. For example, reverberation can be reduced by considering the energy diffusion effect of large components of speech energy. To this end the authors of [37] used temporal filters to modify the power spectral density functions of the log filter bank coefficients. Enhancing the the feature trajectories for spectral features should yield cepstral features with better feature statistics.

Our understanding of the human hearing process motivated the use of filter banks in ASR front-ends [38, 39]. Furthermore, using a filter bank representation to represent signal energies proved a significantly robust representation. In fact, by using this representation

MFCCs could remain the most widely used features of ASR applications for two decades [40]. MFCC features have excellent discrimination capabilities and low computational complexity. It might be possible to improve further ASR performance in noisy conditions through the clever design of filter banks. The authors of [40] show in their work that a dense, smooth filter bank and some alternative energy estimation schemes seem to be more robust in noisy conditions than conventional MFCC or PLP features. They find that filter bandwidth is more important than filter shape for ASR in noisy conditions. Using a Teager-Kaiser energy estimator in conjunction with Gammatone filters improved their results for most noise types. Apparently it is best to place the Gammatone filters on an equivalent rectangular bandwidth curve. With this design, the results are best for especially the larger filter bandwidths.

2.3.2 Re-shaping feature statistics

To a limited extent, the classification error of HMMs may decrease if we transform the data to fit the assumptions of Section 2.2 better. One way of doing this is to decorrelate the elements of the feature vector (frame) as well as adjacent feature vectors as much as possible. This process allows a computationally far simpler classifier: one that uses diagonal covariance matrices for HMM states. Similarly, dimensionality reduction allows models that are computationally more viable. A discrete cosine transform (DCT) [41] has long been used in ASR front-ends to decorrelate feature channels during the speech-coding process. The DCT does, however, prove inadequate. In [42] Gales aptly states: "*It is hard to find a single transform which decorrelates all elements of the feature vector for all states.*" Malayath and Hermansky [43] also point out that although the DCT works well to decorrelate feature vector elements, it is not designed to preserve the separability of phones. By constrast, using data-driven approaches may be beneficial to reduce dimensionality. Logically, it makes sense to retain only the dimensions that carry the most useful information.

Gales [21] explains that:"*It is possible to use data-driven approaches to decorrelate and reduce the dimensionality of the features. The standard approach is to use linear transformations.*" There are two ways of conducting a linear transform in practice: (1) supervised, using class labels of the feature frames and (2) unsupervised, taking into account only general feature attributes such as feature variances. In [44] the authors compare different types of classes (phone, state or component) for the supervised case. Since the transforms attempt to separate each of the classes as far as possible, the choice of class is crucial. The study shows that using state and component classes works better

than using less specific levels of phone and word classes. It is also true that most systems use component classes to transform ASR features, since these classes best fit the assumption of using diagonal covariance matrices with HMMs [21].

Two closely related linear transformations in pattern recognition are principal component analysis (PCA) and linear discriminant analysis (LDA). These techniques are widely used. Apart from speech processing [44], applications include face recognition, hand recognition, object recognition and robotics [45]. With PCA, an unsupervised transform attempts to maximise the feature variance, selecting components from a specific subspace. The objective of LDA is more specific. Here the aim is to maximise the ratio of between-class variance and the within-class variance for each class label. With LDA, taking the average within-class covariance matrix as a diagonal matrix complements the diagonal covariance of HMM states [21]. Using full covariate class matrices does improve LDA. A heteroscedastic discriminant analysis (HDA) [46] is an example of such use. In speech processing, an HDA is often followed by a semi-tied transform [42] because then the full covariance matrix does not decorrelate the feature vector elements. Another commonly used variant of HDA is the heteroscedastic LDA (HLDA) transform [47]. It differs from LDA by taking into account all the dimensions of the feature-space, before discarding the dimensions not retained for recognition. When using diagonal covariance matrices with HMMs, HLDA provides the best feature mapping [21].

Finally, feature correlations can always be captured by using full covariance matrices to represent the Gaussian distributions of each particular HMM state. This is generally impractical, however, since the size of such a model set is simply too large to be accommodated in LVSR systems [21]. Another drawback is that a large number of parameters per Gaussian component will limit the number of components that can be estimated robustly. Using multiple Gaussian components (mixture models) can alleviate the number of parameters and add some of the benefits of a full covariance matrix. Not only the non-Gaussian state distributions are better represented, but also the correlations. Consequently, estimating mixture models for HMM states is common practice [48]. A semi-tied transform (also called a maximum likelihood linear transform [49]) is a go-between solution that effectively enables a few "full" covariance matrices to be shared over many distributions [42].

2.3.3 Improved model training

Large vocabulary systems require huge numbers of HMM parameters, which can be a problem to train effectively. In particular, it is challenging to ensure sufficient estimation of all these parameters with limited data. This section discusses two possible ways of

alleviating the challenge of this estimation. Firstly, sharing data among different data sets is helpful, but so are improved training schemes (should the current training method be suboptimal). In fact, work in both these areas shows improvement of HMM-based accuracies. Adaptation techniques share data by updating the existing HMM parameters with the estimates obtained from other (similar) adaptation data sets. Section 2.3.3.1 contains a description of the most frequently used adaptation approaches. We end our discussion in Section 2.3.3.2, where we explain the discriminative training techniques that achieve even better model estimates.

2.3.3.1 Adaptation

One could use MAP estimation merely to add more prior information to an HMM training process. An important assumption of the maximum likelihood estimation method is that all parameters are estimated for a sufficiently large data set [50]. Since this is the estimation method for standard HMMs, it is challenging to ensure the robust estimation of all parameters. The complexity of the speech signal with all the variability sources that play a role in the estimation process simply leads to large data hunger [51]. Given the right set of adaptation data, MAP adaptation can alleviate this problem to some extent. With MAP adaptation, one usually adapts mean estimates of the HMM states, but it is also possible to adapt all other HMM parameters [50]. A problem with MAP adaptation is that the parameters of HMM states only update when there are examples in the adaptation data. In particular, adapting models with limited data is not an effective strategy [52] for large-vocabulary speaker-independent systems containing vast numbers of parameters.

There are alternatives to MAP adaptation for this purpose, which establish broader relationships among the HMM parameters of the training data, enabling better parameter updates from the same set of adaptation data. In [53], the authors introduce a regression-based model prediction approach. Similar to MAP, this adaptation technique still uses a Bayesian approach to combine parameter estimates with new predictions. In essence, here the small number of well-adapted parameters predicts improved parameter values for parameters that are unseen or poorly modelled using the adaptation data. Model transformation is another approach that allows information about the more general acoustic environment to be added to an existing set of HMMs. Provided that the relationship between a parameter and the adaptation data can be established, the parameter can be updated successfully. The same transform then also enables many model parameters to be updated, even when those parameters have not been seen in the adaptation data. Many speech recognition systems use the maximum likelihood linear regression (MLLR) transformation for this purpose.

The main objective of the MLLR adaptation technique is to find a set of transformation matrices which, together with HMM parameters, maximise the likelihood of adaptation data, given the model set. Initially, updating only the mean parameters using MLLR adaptation provided HMMs for a range of speaker adaptation tasks in speech recognition. The assumption was that the main differences between speakers were in fact characterised mainly by the mean parameters [54]. Later applications followed for speaker adaptation and environmental adaptation. For instance, using MLLR in an iterative, unsupervised fashion proved useful to reduce mismatch in noisy data [55]. It quickly became apparent that not only the mean value parameters but also the variance parameters of HMMs required updates [52].

In practice, the MLLR transform applies in different ways to HMM-based speech recognition models. In [56], Gales presents the two commonly used forms. Where one estimates separate mean and variance parameter transforms for an unconstrained MLLR transform, the mean and variance parameters update, using the same single transformation matrix for the constrained MLLR (CMLLR) method. The CMLLR transform is better suited to environments where speakers or the acoustic data change rapidly. Only the transformation parameters update and the model parameters are left unchanged, which is not the case for the unconstrained MLLR transformation.

The MLLR and MAP adaptation techniques may be combined so that prior knowledge constrains the MLLR transform. In this case, the MLLR algorithm includes a Bayesian criterion to update transformation parameters, instead of plain maximum likelihood estimation (MLE). The new method is called MAP linear regression (MAPLR) adaptation. Since a large number of prior densities remain an estimation problem with MAPLR, the authors of [57] take a structural MAPLR (SMAPLR) approach where they organise prior density estimation with a hierarchical tree-based structure. This representation improves prior density estimation.

In [58], Nakano et al. implement a CSMAPLR transformation for an HTS system, which is a combination of CMLLR and MAPLR techniques. They use adaptation with very limited speech data to build a TTS voice successfully for a target speaker. They conclude that using the CSMAPLR transform provides TTS voice characteristics closer to those of the actual target speaker than using CMLLR or MAPLR techniques on their own. In fact, in view of the exciting possibility of creating different TTS voices by simply applying model adaptation techniques, the authors of [15] write: "*The attractive part of an HTS system is that voice characteristics, speaking styles or emotions can easily be modified by transforming the HMM parameters using various techniques such as: adaptation, interpolation eigenvoice or multiple regression.*"

The success of the adaptation and interpolation approaches does rely, however, on the quality and quantity of the adaptation data. Given the sheer number of spoken languages in the world, the availability of high-quality speech data is a luxury. When training data is limited, other approaches are helpful in improving model estimation. Accordingly, the next section discusses the approaches that improve the training algorithm itself. Section 2.5 also describes various methods to extend, specifically, the limiting factors in training data.

2.3.3.2 Discriminative training

Training HMM-based speaker-independent (SI) systems on studio quality speech data leaves two important types of variability sources inherent in the speech signal. These are phonetic variability and the variability among different speakers [59]. What makes these types of variability unique is that the sources of these two variations are, in a sense, independent of the information content of the rest of the speech signal. The core motivation for speaker-adaptive training (SAT) approaches stems from these observations. Minimising the variability introduced to SI systems by varying pronunciations of different speakers should reduce recognition error. SAT can be performed by using any of the adaptation techniques mentioned in the previous section (see 2.3.3.1). By estimating speaker-specific transforms, it is possible to map speaker-dependent models to a central, more compact model representation of an SI system.

The MLE approach commonly used for training HMMs has its shortcomings. Firstly, if unlimited data were available and secondly, if speech had the exact statistics assumed by a particular HMM definition, then MLE would be the optimal estimation criteria [60]. Since this is clearly not the case in the real-world scenario, alternative criteria may be found to optimise acoustic models and predict unseen test data in this environment better. One such set of alternative criteria is the estimators used for training discriminatively. A difficulty with these methods is that they are not guaranteed to converge under all practical conditions [61]. Therefore, much energy has been spent on trying to develop parameter estimation techniques for discriminative training criteria that can achieve fast and reliable convergence. Another problem is that discriminative criteria can overtrain, requiring techniques to improve the generalisation of the training data [21].

In [21] the authors give a summary of the different criteria commonly used for discriminative training. Among these, the main criteria used in speech recognition experiments are: (1) maximum mutual information (MMI) [62]; (2) maximum classification error (MCE) [63]; and (3) variants of minimum Bayes' risk (MBR) training. With MMI,

the objective is to maximise the mutual information between a word sequence and the information that the recogniser extracts from the feature observations. MMI is one of the first criteria used for performing discriminative training [60]. MCE also considers the word sequence. Here the difference between the likelihoods of the correct word sequence and all other competing word sequences is smoothed by using a sigmoid function as a metric. MBR works by minimising the expected error of recognition. There are many error (loss) functions, which can approximate recognition errors. Typically, the functions are based on a word, phone or even phone frame error units. Researchers use minimum phone error (MPE) training [64] and minimum phone frame error (MPFE) training [65] for LVCSR, owing to better generalisation support. Using a word error rate metric does not work well, since not all word sequences will be seen during training. MPFE attempts to weigh the estimation of the expected loss correctly, given the number of frames associated with the phone units.

Discriminative training initally led only to improvements in the systems trained on limited amounts of data. The complexity of LVSR systems (computationally as well as the numerous parameters) makes it difficult to train discriminatively. Valtchev, Odell, Woodland and Young [66] manage to reduce computational complexity for their algorithm training with the MMI criterion. The authors also carefully employ the splitting of mixtures in every HMM state to obtain the right balance between parameter estimation and the amount of training data. A comparison of MMI, MCE and MPE criteria on the Wallstreet Journal corpus shows that they all significantly outperform ML training; that MCE and MPE are better than MMI training and that MPFE provides small but consistent gains over MPE [21].

Many attempts have been made in the past to use feed-forward neural networks (NN) as a replacement for GMMs. There are good reasons for these attempts. In [67] the authors explain that NNs do not require the same detailed assumptions about feature distributions as HMMs do. Furthermore, NNs provide a simple way of combining widely differing features (including continuous or discrete types) and they make far better use of the speech data to constrain parameter estimation. Recently, the discovery of a far better way of training NNs sparked renewed interest in applying these classifiers to speech-processing tasks. Initialising the weights for the NN by using a deep belief net (DBN) and some discriminative fine-tuning creates a DBN that trains faster and overfits the data far less. In [67], using speaker-adaptive and discriminative features as input into DNNs improves phone recognition accuracies for the TIMIT corpus even further. Since DBNs potentially provide a much more dense representation of the speech data than HMMs do, they are an attractive predictor to use. Current work continues to analyse the properties of DNNs that lead to the improved modelling of speech data [68, 69].

LVSR remains a challenge, also when adding DBNs as predictors. In [70] a training recipe is developed for using the DBNs for this task. By training HMMs discriminatively, Woodland and Povey [60] have further refined their approach and successfully shown how even for LVSR it is possible to obtain significant reductions in error rates with discriminative training. Interestingly, they obtained these improvements for a conversational data set and when they used context-dependent HMM parameters, such as triphones and quinphones. As mentioned in Section 2.2.1 above, more recent work includes DNNs so that the DNN models the observation likelihoods for each (context-dependent) HMM state [24]. The need to use context-dependent models together with the now even more powerful predictors emphasises the importance of contextual effects. It seems that accurately representing the phonetic variability with fewer parameters is simply not possible yet. Next we describe models that attempt to improve context modelling (Section 2.4).

2.4 Modelling contextual effects as trajectories

Many factors influence the realisation of a speech signal. The previous sections in this chapter describe many advanced methods to model this variability and emphasise the progress made with achieving accurate and robust speech representations. These approaches work well for large datasets. Modelling speech with limited data still requires more innovative ways of improving the accuracy of recognition. A detailed understanding of all the variations involved has to date inspired many modelling advances. The summary in [51] gives a perspective of the variation encoded in speech and the associated challenges. After reducing signal mismatches (see Section 2.3.1), ensuring better parameter estimation (Section 2.3.3.1) and even using more powerful predictors (Section 2.3.3.2), speech systems still have to maintain extensive inventories of context. This section describes how trajectory models provide an improved representation of the different contexts that various contextual effects create.

Sim and Gales [71] employ a trajectory model aimed at reducing the limitation of the conditional independence assumption on HMMs. Related to a feature transform, these models work by adjusting state-specific parameters. As a result, the mean and covariance parameters also become dependent on the position of the current frame in time. A complete definition of the new model fits the description of a minimum phone error training of the feature space (fMPE) model. fMPE is an established discriminative training criterion [72]. In [73], Sim and Gales describe how they use fMPE as a time-varying model.

At the time of writing, [71] reported that various attempts had been made to model contextual effects explicitly as phone transition trajectories. The authors also wrote that using ML training and attempting LVCSR with these approaches yielded mixed success. Techniques such as trajectory HMMs [74] and stochastic segment models (SSMs) [75], are employed by incorporating explicit trajectories into an HMM framework or by defining longer-term segmental models of variable length within this structure. Switching linear dynamical systems (SLDS) is an extension of the SSM, using a standard linear dynamical system to model the sequence of the observations within a segment. These segments are assumed to be independent, but this is a non-optimal modelling assumption owing to the co-articulation effects among segments. The SLDS approach improves this by propagating the posterior distribution of the state vector over segment boundaries [76].

The lack of adequate temporal modelling has kept resurfacing ever since then. When comparing TTS and ASR modelling techniques, the authors of [17] write about the importance of feature dynamics for speech synthesis. TTS requires explicit relationships between dynamic and static features and implements such relations as an inference of an observation vector approach. For the same purpose, trajectory HMMs are also incorporated in similar work [35]. It is clear to [17] that context modelling is a major problem when acoustic data is limited and the author writes about trajectory models: "*Implementation of such statistical modelling frameworks for ASR and TTS also requires consideration of the sparse nature of contextual modelling, where some models, such as switching linear dynamic system, are able to provide implicit handling of co-articulation effects resulting in a more parsimonious model, while others constitute a more direct extension of the conventional HMM framework and necessitate a reformulation of parameter tying algorithms.*"

In a novel approach, the authors of [4] introduce the hidden trajectory model. A bi-directional filtering of the vocal tract resonance target sequence enables them to implement a variable length representation of long-contextual-spanning speech effects. By adding a likelihood score computation and an A*-based time-asynchronous lattice-constrained decoding algorithm, they succeed in obtaining significantly high phone accuracy for the core test set of the TIMIT corpus [1].

2.4.1 The role of frame-based features

A motivating factor for segmental models is that these models may exploit the acoustic features that reside at the segmental, rather than at the frame level. The encoding

of typical ASR and TTS features, however, forms separate frame-based units. An alternative type of segmental feature is one that simply represents the time evolution of frame-based features by some parametric description [6].

Trajectory models, in effect, smooth the features at the frame level. In this regard, they are related to the low-pass filtering commonly used to increase the signal-to-noise ratio for noise-robust speech recognition techniques. The goal of noise-robust approaches is to systematically "recover" corrupt speech frames. In principle, if reliable features can be identified, these can in turn be used to make more accurate predictions about less reliable ones. Section 2.3.1 reports on the work of Chen and Bilmes [36] who use ARMA filtering to smooth the cepstral features. If over-smoothing occurs, the more definite boundaries of speech events can be modelled by using edge-preserved filtering [16]. Xiao and Li [37] show that besides normalising the probability distributions of speech features, the temporal characteristics of the feature trajectories can also be enhanced at the spectral level. Consequently, ARMA filtering may be a useful preprocessing step prior to fitting trajectories.

Gales and Young describe how HMMs are generative models, and that it may also be of interest to consider how well these models can generate speech. The main consideration in this approach is whether dynamic features do enforce accurate trajectories for static parameters. The conditional independence assumption of HMMs implies that feature trajectories will be piecewise stationary. It is questionable whether these trajectories would fit real speech sufficiently, where spectral trajectories vary more smoothly [21]. Nevertheless, the prospect of generating more examples if the available speech data is limited, is interesting. The next section introduces techniques that build on this idea.

2.5 Augmenting limited training data

Inadequate parameter estimation usually occurs when one tries to train speech systems for under-resourced languages. There is simply not enough data to cater for all the co-articulation effects these systems have to model. Furthermore, the cost of creating corpora that would suffice remains a major impediment. This condition and the need for speech systems to be developed in increasing numbers of languages sparked a renewed interest in research, which may alleviate the current situation. Data augmentation schemes aim to increase the data available to a speech system artificially [77].

2.5.1 Semi-supervised training

Much progress can be made with increasing the training data by using a semi-supervised training strategy to create transcriptions for untranscribed data. Large numbers of speech recordings often do exist in a low-resource language, for example, broadcast news data. In [78] a complete real-time broadcast news system is built where additional lightly supervised training data, advanced acoustic models and finally system combination yield large performance gains. A diarisation step segments the audio, clustering blocks of the same speakers and possibly similar acoustics. Then these clusters of acoustic data are used for building all the different speech systems, and are later combined to produce an improved result. For the broadcast news data that they have, only very rough text is available for the "untranscribed" data and consequently extensive attempts are made to recover additional information from these texts.

Qian and Yu [79] explain that four types of speech corpora are usually found for low-resourced languages. The first type is the limited (accurately) transcribed speech data. Secondly, there may be a larger set of untranscribed data for the same target language. Data may be borrowed from related languages, which may also include manually transcribed as well as untranscribed speech corpora. For some of the languages, the manually transcribed corpora may be a rich source of additional phonetic material. Standard speech systems are usually built on only the first (accurately transcribed target language) set of data. Consequently, several strategies have been proposed to borrow data from each of the other data set types.

2.5.2 Synthesising the data

The key property of data augmentation is that these approaches produce some sort of synthetic data. Therefore, a useful way to group data augmentation schemes is to consider closely what type of data a technique produces. These types also play a significant role in selecting or "fine-tuning" a data augmentation scheme. In [77], the three data types referred to are other language, unsupervised and synthesised data.

Sometimes abundant high-quality data is available for a language closely related to the low-resourced target language. Borrowing examples at a contextual level may prove beneficial in these cases, and would require the use of multilingual systems. These systems use multilingual acoustic models trained on universal phone sets [80] to exploit resources across language barriers. Advantage can also be taken of multilingually trained SGMMs or NNs as a data-driven approach to mapping the acoustic information in a cross-lingual mode. A drawback of this approach is that it is often not clear how to obtain sufficient coverage for context-dependent phones with universal phone sets.

The unsupervised category is mapped directly to the above-mentioned roughly transcribed or untranscribed data sets. Here the steps of bootstrapping, filtering out "dirty" utterances using a confidence threshold, and re-training the systems only on selected utterances describe the general semi-supervised procedure. Vast amounts of this data might be collected, but the nature of unsupervised data restricts its use with advanced acoustic modelling approaches such as discriminative training that is sensitive to transcription quality.

Synthetic data may refer to perturbed data or entirely new examples of contexts, which have been artificially generated. The great advantage is that this technique can generate vast amounts of data. The main problem to be solved is the quality of synthetic data.

2.5.2.1 Perturbed speech data

There are many ways to perturb speech data. Vocal tract length perturbation (VTLP) [81] is a technique that adjusts the vocal track length during standard feature extraction (generating MFCC or PLP features). The original motivation for VTLP was to learn a larger number of robust multi-layer perceptrons so that these would be less affected by changes in the vocal tract length of different speakers. Moreover, the speech of a speaker could be altered to represent more closely the speech of another speaker, using a CMLLR transformation. By training the transformation itself on the statistics of the speaker to whom the utterance belongs, one can borrow data for a particular speaker from a range of alternative speakers. However, applying a single global transform may lack the capability to map the individual differences in pronunciation among speakers accurately. We think it may be possible to capture and transform these differences more precisely by using some form of trajectory model.

Stochastic feature mapping [82] attempts to find a mapping function so that a set of features may be statistically converted between two speakers. During this process, Cui, Goel and Kingsbury apply a feature space MLLR to reduce speaker variability. Their goal is to train LVCSR using DNN acoustic modelling and to report improved recognition performance after cross-entropy (CE) and state-level MBR training.

Apart from VTLP, the authors in [83] continue to experiment with speech rate distortion and frequency-axis random distortion in their quest to generate synthetic training samples. Randomly varying the speech rate within specified limits generates synthetic samples for speech rate distortion. With frequency-axis distortion, random oscillations are introduced for each time and frequency bin extracted from the spectrum of the speech signal. Interestingly, the gains these authors obtain for the three above-mentioned distortion methods almost seem to be additive. It may be that each of the distortion criteria

successfully introduces different types of variability not seen in the low-resource data set. They claim that DNN-HMM systems seem to be the preferred method for speech recognition applications in low-resource languages, since their GMM-HMM system requires far more data to achieve similar accuracies.

2.5.2.2 Speech synthesis

The previous section describes ways of creating synthetic examples in a random fashion, usually following some kind of distribution to guide the process. Even more detailed strategies may be developed. The term speech synthesis becomes more appropriate to describe the processes that generate such tightly constrained synthetic examples of context. Consequently, the speech synthesis approaches play an important role in this area. A novelty of speech synthesis is that it can generate entirely new synthetic examples of speech because concatenative synthesis combines existing waveform segments to generate the synthetic utterance.

A more interesting possibility is, however, to generate and concatenate waveform segments using statistical models (HMMs, which produce the speech parameter sequences) [15, 19]. Since these models are trained on normal MFCC or PLP features, synthetic examples may also be constructed in feature-space. Speech synthesis offers more flexibility to generate synthetic data than the previously discussed techniques. Entirely new utterances for unseen contexts may be synthesised, given an arbitrary transcription to guide the process. It is also possible to condition the HTS models for a particular type of channel condition and synthesise examples to simulate a particular environment [84]. In this way, it may even be possible to target only specific confusions.

2.6 Conclusion

Based on the discussion in this chapter, we conclude that the acoustic modelling of speech systems requires more research, especially for low-resource languages. Combining data augmentation approaches with context modelling may alleviate some of the data shortage of these systems. The next chapter begins with a more detailed analysis of the co-articulation effects that form the primary motivation for the approach adopted in this thesis.

Chapter 3

Co-articulation: an initial analysis

3.1 Introduction

Multiple sources of variation in speech data (the speaker, channel conditions, speaking styles and co-articulation) complicate ASR system development, leading to a requirement for large corpora of training data (Chapter 1). It is not entirely clear why so much data is required to train accurate systems. As discussed in Section 1.1, the phone recognition accuracies in different subsets of the TIMIT corpus indicate that context modelling (and therefore co-articulation) presents significant challenges to current modelling techniques.

As an initial analysis, we attempted to investigate this co-articulation effect. The focus was on the ASR front-end, since all the acoustic information an ASR system requires is encoded in the features that are extracted from the speech signal. These features can be analysed at the frame level. We could not directly observe the co-articulation process on the individual frames. Instead, multiple examples of the same units of speech (phones) allowed the calculation of the distance of a feature vector from a reference unit estimate. These reference unit estimates are the mean values obtained across all feature vectors for a particular phone label. As one phone transitions into the next, the distance measurements of the individual vectors are expected to diverge from the reference unit estimate. These reference units are expected to act as if they were "targets", with some form of transition occurring from one target to the next over time. In this way, a series of frames (for a segment of speech) can describe intra-transitional co-articulation. In essence, the core technique used in the analysis described in this chapter identifies a reference unit for every phone, and then track the trajectory by which the audio signal diverges from these reference values over time (Section 3.3). The separability of reference unit estimates is also tested using a phone identity separation test (a technique where the

distance between the two unit estimates for the particular phone transition of a specific segment is calculated and used to determine the phone boundary), which is described in Section 3.3.2.

The trajectories that are formed can be analysed for each individual transition. Characteristic trajectory-based parameters may be extracted to describe different types of co-articulation. We were specifically interested in determining whether different types of phone transitions occurred, and whether similar transitions were observed over similar phone classes across multiple speakers. Section 3.4 describes how the ability of these parameters to predict the ASR boundaries was investigated. Grouping multiple transitions with regard to broad phonemic classes also allowed us to show how co-articulation differed for certain classes.

3.2 Terminology

Table 3.1 gives definitions of the frequently used terms introduced in this chapter.

Term	Interpretation
Frame	The window of time over which the features are calculated for particular time instants.
Feature vector	A multi-dimesional vector of feature values calculated for a single frame.
Frame level	A frame-based representation of the speech data.
Segment	The time period over which a single transition from one acoustic unit to another is modelled. An acoustic unit is typically modelled as a monophone, biphone or triphone.
Phone transition	The acoustic change of one acoustic unit to the next within a segment.
Reference unit estimate	The mean value estimated across all frames for a particular acoustic unit.
Phone identity separation test	Verifying the separability of phone identities within a segment by using the selected reference unit estimates.
Phone identity	The label assigned to the frames of an acoustic unit.
Characteristic parameters	Trajectory-based parameters that fully describe a complete phone transition.

TABLE 3.1: *Terminology used throughout this work.*

3.3 Tracking co-articulation

Typical ASR systems use features such as MFCCs, PLPs or features derived from them, such as bottleneck features, to encode the speech waveform. The audio is sampled at a specific rate, generating a vector of values for every particular time instant (frame). Any co-articulation affecting the accuracy of systems built from these features is therefore expected to be present too in the frame-based representation. Determining the impact

of co-articulation on a specific segment of speech would prove difficult. The reason is that the magnitudes of each feature value and every feature vector are altered by a speech production process that one is unable to observe directly, given only the speech data. There is a solution for the problem. Analysing multiple examples of similar phone transitions can shed light on the relative differences between them.

The same holds true at the frame level. Given a sufficient number of frames for a unit with a particular acoustic quality (such as a phone), a fixed unit reference can be estimated. Individual frames can then be categorised in terms of these unit estimates, by measuring the difference between the actual value and the unit estimate value per feature vector (Section 3.3.2). We analysed these differences for the feature vectors of a segment to inspect the influence of co-articulation on a per-frame basis. It was necessary to assess the difference measures so as to ensure that we were constructing meaningful representations of the modelled acoustics. In essence, given the specific unit mean variables (corresponding to the phone labels of a segment), the tracked contextual differences (trajectories) must yield the best possible separability of the frame-based features to the left and right of a phone transition boundary. Some phone transition classes have such strong co-articulation effects that separation is not fully possible. This is typically the case for closely similar-sounding phones. Furthermore, we expected the unit mean variable representation to be less accurate, especially for unvoiced phone classes. We chose to ignore issues such as transitions that are not smooth or phones not having such stable regions as the vowel phone classes for this initial exploratory analysis.

Section 3.3.2 explains how we used class separability to evaluate the overall accuracy of the technique. Meaningful trajectories could be extracted from the difference values by using polynomial functions. The different ways in which these trajectories categorise different types of acoustic change were investigated, as reported in Section 3.3.3.

3.3.1 Experimental data and segmentation

The well-known TIMIT speech corpus [1] was used for all the co-articulation experiments presented in this chapter. The corpus consists of 630 speakers from eight major dialect regions in the United States. Each speaker read 10 utterances, totalling 6300 utterances. The corpus is divided into a standard training and testing set. There were 326 male and 136 female speakers for the training data, totalling 462 training speakers. The types of sentences read were divided into three parts: *sx*, *si* and *sa*. MIT designed the 450 phonetically balanced *sx* sentences, and the *si* sentences formed 1 890 phonetically diverse sentences designed at SRI. Finally, the test set consisted of 168 speakers, selected so that no speaker was included in both the training set and the test set.

In order to generate automatic alignments, we built a standard HMM-based ASR system, trained by using the training set of the TIMIT corpus. The system employs a context-dependent cross-word phone recogniser with tied-triphone models for this purpose. It uses 39 MFCC features, including 13 MFCCs and their first and second-order derivatives. MFCC parameters include a window size of 25 ms and a frame rate of 10 ms respectively. Cepstral mean normalisation is applied. With regard to the modelling structure, each triphone model has three emitting states with seven Gaussian mixtures per state and a diagonal covariance matrix. We generated state-level phone alignments, using the system in forced alignment mode. These alignments provided the HMM-based phone transition boundaries used in the rest of this chapter.

3.3.2 Estimating a context-dependent unit reference and frame-based differences

As mentioned in Section 3.3, the key to analysing co-articulation effects was to compare the individual feature vectors (for specific frames) with their respective unit estimates. To this end, we first determined a unit estimate based on phone identity, and then measured the deviation from this measure on a per-frame basis. In short, the process can be described by the following steps:

1. Normalise all feature vectors to have zero mean and unit variance for each cepstrum, across all the frames in the data set.
2. Use ASR alignments and phone labels to cluster frames of the same phone identity.
3. Estimate the mean feature vector for every cluster of frames to create a fixed unit reference.
4. Calculate a difference measure to characterise the phone identity for every feature vector, using the correctly selected reference unit estimates. (These values will describe the closeness of the feature vectors to the unit estimate.)

ASR alignments were used to associate frames with the corresponding phone labels, leading to a segmentation of frames that would normally be selected to model these same labels during the ASR training process. The mean of all feature vectors for every monophone in the training corpus was calculated. Different means can be created: summing over phones of all speakers or only for the same speaker. In addition, either all the frames of a monophone or only the central frames (associated with the centre states of HMM alignments) can be used.

Various analytical functions may be used to calculate the extent to which each feature vector diverges from the respective reference values. We experimented with three measures, namely the Pearson correlation coefficient, the Euclidean distance and the dot product between two feature vectors. These measures can be defined as indicated below. (Appendix A lists the definitions of all the metrics in this thesis.) Let X and Y be any two random variables. Then the Pearson correlation coefficient is given by:

$$r(X,Y) = \frac{\sum_{i=1}^{N}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{N}(X_i-\bar{X})^2}\sqrt{\sum_{i=1}^{N}(Y_i-\bar{Y})^2}} \tag{3.1}$$

where for N dimensions, $\bar{X}$ and $\bar{Y}$ are the means of each of the random variables X and Y respectively. Secondly, the Euclidean distance for the two input vectors X and Y is defined as:

$$d(X,Y) = \sqrt{\sum_{i=1}^{N}(X_i-Y_i)^2} \tag{3.2}$$

where X_i and Y_i are the separate dimensions of the N-dimensional input vectors X and Y. Finally there is the dot product:

$$X \cdot Y = \sum_{i=1}^{N} X_i Y_i \tag{3.3}$$

On the basis of a single phone transition, a phone identity separation test could be performed. The average difference between each unit estimate and the feature vectors from the left and right of a (known) phone transition boundary was used for this evaluation. Achieving a separable phone identity result required a smaller average feature vector difference (for both monophones): the matching label and unit estimate pair of the particular transition were compared, on both sides of the phone transition boundary, with the measurement for this label and the unit estimate of the second phone label. Table 3.2 indicates the number of phone transitions for which both phones can be separated, using the various difference measures described above. It was found that though all three difference measures provided fair class separability, the Euclidean distance outperformed the correlation and the dot product.

Difference measure	**# Separable identification**	**% Accuracy**
Euclidean	40 558	77.1
Correlation	39 190	74.5
Dot product	36 644	69.7
Euclidean (ASR centre)	42 747	81.3
Correlation (ASR centre)	37 585	71.5
Dot product (ASR centre)	31 074	59.1

TABLE 3.2: *Number of phone transitions for which phone identities can be separated using mean frame-based values and known ASR boundaries. Centre state-level phone alignments provided even higher separability (ASR centre). In general, the Euclidean distance outperformed both the correlation and the dot product.*

The state-level phone alignments (Section 3.3.1) can be used to evaluate separability only for the feature vectors associated with the phone centres. Switching to these state-level boundaries (indicated as ASR centre in Table 3.2) resulted in an even greater improvement for the Euclidean distance, but not for the other measures. This shows the presence of two opposing effects: (1) stationary components at phone centres and (2) the encoding of co-articulation in the unit estimate variables. At the phone centres less co-articulation yields more separable trajectories, while longer trajectories are likely to reveal more information on the particular phone.

The observed phone identity separation accuracy (averaged over all phone classes) of 81.3% when using the Euclidean distance as difference measure is surprisingly high, given the simplicity of the classification technique. In the remainder of this chapter, we report the results obtained by using the Euclidean distance. Similarly, we focus on the use of speaker-independent monophone means as unit estimates. One expects these speaker-independent monophone means to be more robust because of the larger amount of data available for these units in the training corpus. Calculating a complete set of results, given the different options of unit estimates, indicates only slightly better detection for speaker-specific means or means based only on central frames (see Table 3.2).

3.3.3 Calculating co-articulation trajectories

Co-articulation effects are expected to be tightly coupled to the identity of the monophones involved during any particular diphone transition. Consequently each of the measurements in Section 3.3.2 was used to obtain two discrete values per frame, measuring the difference from the two unit estimates on either side of the phone transition

boundary. In order to create a trajectory from the difference values, we fitted a polynomial function, using least-squares estimation. This approach effectively minimised the squared error (SE) of diphone segments $SE_{segment}$, given by:

$$SE_{segment} = \sum_{f=1}^{F} SE(p_f, \hat{p}_f, \omega) \text{ where } \omega = \bigcup_{f=1}^{F} (p_f, \hat{p}_f) \tag{3.4}$$

where for a specific diphone segment, the sum of the squared errors between the trajectory value and the value of the difference measurement at each frame f is computed for the set $\bigcup$ of parameters ω, which includes all the frames F of the segment.

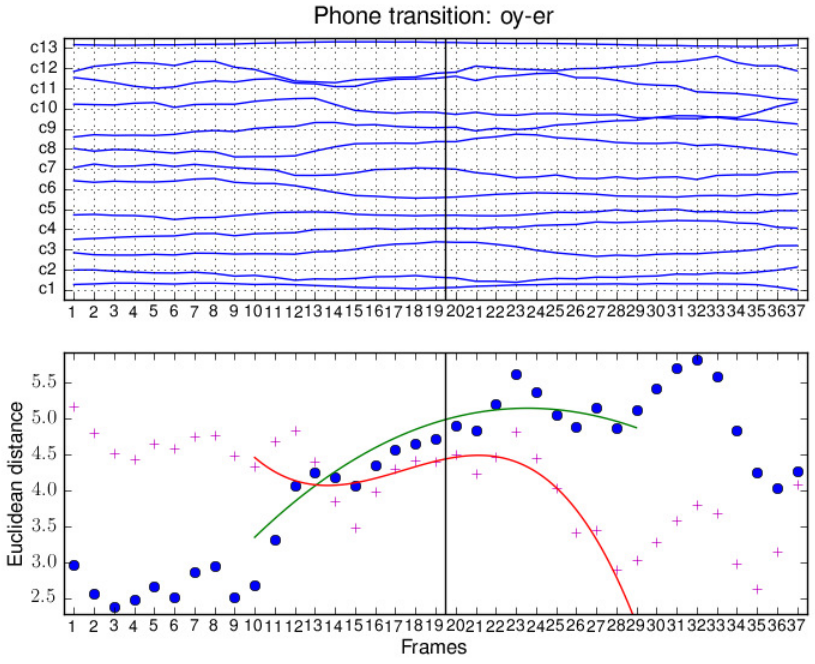

FIGURE 3.1: *Gradual trajectories (bottom) and MFCC frames (top) revealing strong co-articulation for the vowel-vowel phone transition.*

The order of the polynomial is an important factor to consider, since higher-order polynomials quickly lead to over-fitting. We describe the trajectories formed near phone transition boundaries in terms of a third-order polynomial function and we only fitted the frame sequence closest to the boundary. This was done to prevent interference from additional co-articulation to the left and right of the phone transition being analysed. Only the closest 50% of monophone frames were used in the experiments, effectively describing a diphone, a heuristic measure meant to obtain a balance between including only the relevant part of the trajectory and still retaining sufficient frames for analysis.

In order to model a phone transition, two trajectories (one using each reference unit estimate as target) were constructed.

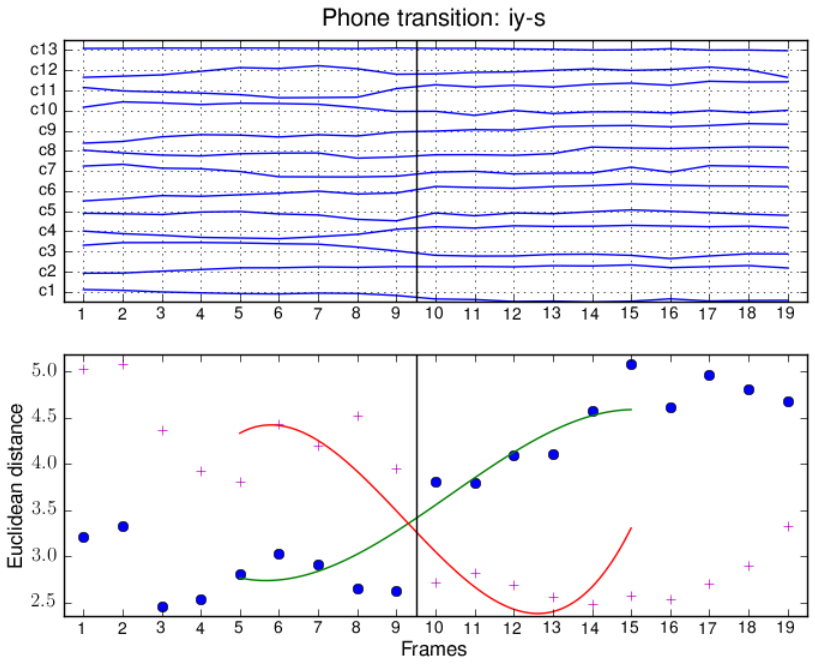

FIGURE 3.2: *Steep trajectory slopes (bottom) and changing MFCC values (top) revealing the definite transition of the vowel-fricative class near the ASR boundary and co-articulation effects flowing well into both phones.*

To demonstrate the prominent types of co-articulation observed, we present four figures as examples. The plot shows the stacked 13 MFCC coefficients for all the frames of the monophone transition, the Euclidean frame-based difference values and the final diphone trajectories consisting of the two polynomial functions. The intersection of the polynomial functions estimates a trajectory-based phone boundary, which is sometimes accurate, or sometimes to the left or right of the ASR boundary. Blue dots indicate the difference values for the first phone; similarly, the red crosses correspond to the difference values for the second phone label. The phone transition boundary as identified by the HMM-based ASR system is indicated as a vertical line.

Figure 3.1 represents an example of the phone transition /oy/-/er/ within the vowel-vowel class, spoken by a male. Strong acoustic co-articulation over a relatively long period of time is clearly visible for frames 11 – 27. This results in a gradual change and small slope values at the ASR boundary. One can see from the difference values that the mean value can still be classified, assisted by the long duration of the speech segment.

An example of a female /iy/-/s/ transition belonging to the vowel-fricative class is depicted in Figure 3.2. The MFCCs and the difference values clearly show a definite phone transition around frame numbers 9–10, indicating a large difference in acoustic

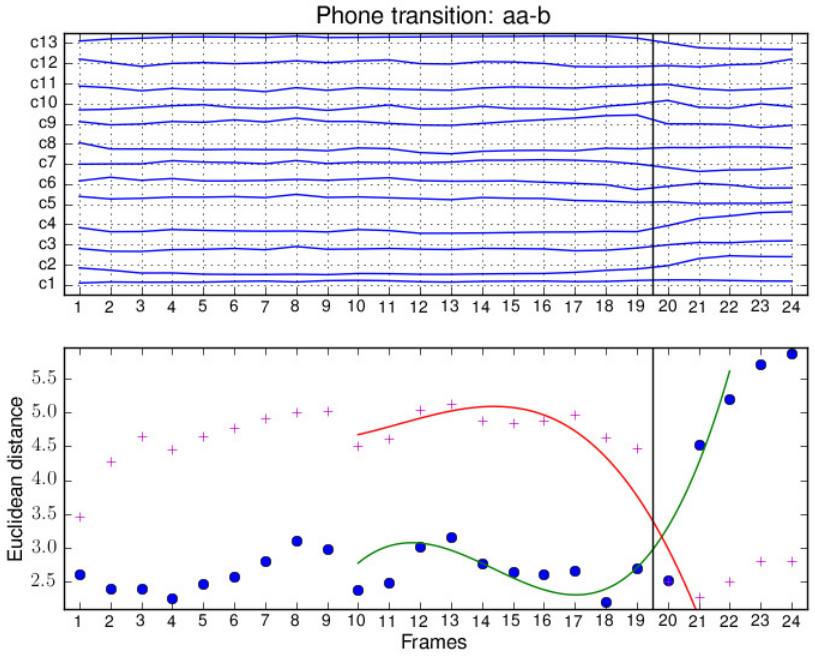

FIGURE 3.3: *Abruptly changing trajectories (bottom) and MFCC values (top) of the vowel-stop class showing little co-articulation effects (affecting only four frames).*

quality between the two phones. It is interesting to note that even in large acoustic change, the co-articulated effects flowing well into both phones are present. Diphone polynomial trajectories have steep slopes at the ASR boundary, and the phone identity separation with regard to the average of the difference values is accurate.

There are also abrupt transitions, with little co-articulation visible. A clear example comes from the vowel-stop class (/aa/-/b/). The MFCCs and difference values both show rapid change in a short period. The co-articulation effects with regard to this transition are seen to affect only four frames: 18–22, and the difference values have high separability (see Figure 3.3).

In all the analyses (also see Section 3.4.2 below) the nasal-nasal transition class tended to be problematic. The MFCC values shown in Figure 3.4 indicate that the straight lines have a closely similar acoustic quality for most of the frames, and the changes are only gradual. The difference values support this finding, showing only gradual transition and poor separation. Co-articulation can be seen to be present for all of the frames, although this may be influenced by the similarity of the two targets being tracked. The slopes of the polynomials have the same sign and are closely similar.

This section demonstrated the use of trajectory models to analyse co-articulation by

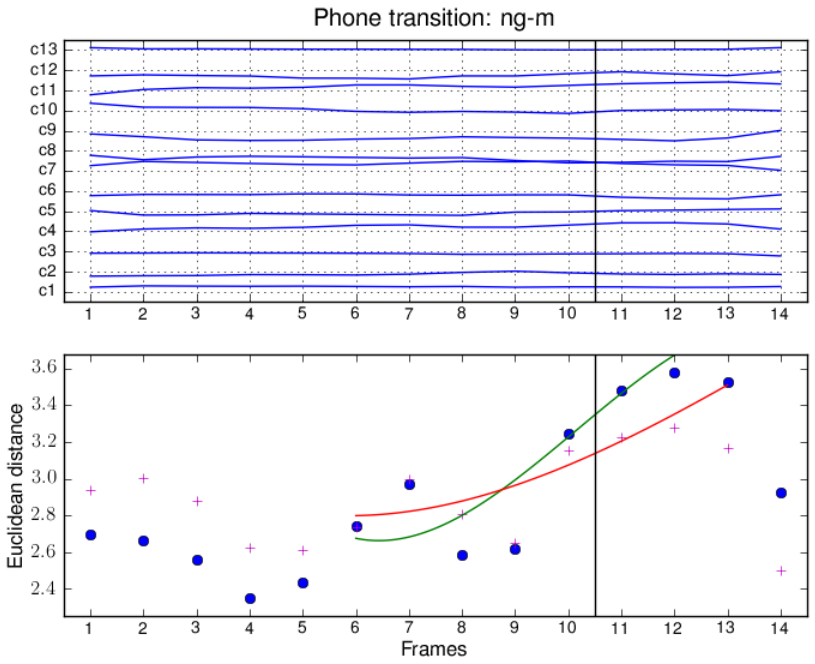

FIGURE 3.4: *Low separability and strong co-articulation effects yield similar MFCC values (top) and trajectories (bottom) for the nasal-nasal class.*

presenting four highly specific examples, which are prototypical of the types of co-articulation observed in the larger corpus. The next section analyses some of these effects over the corpus as a whole, and by averaging over broad phonemic classes.

3.4 Analysis of co-articulation effects

The previous section contains an analysis of co-articulation, tracking the contextual differences on a per-frame basis. The specific trajectory functions that we estimated (for the difference values of a diphone transition) effectively modelled the co-articulation effect for a whole segment. Accordingly, we could show that the underlying speech features are co-articulated; transitional time differences and interacting phone identities strongly affect these frame-based features.

Transitions from one phone to another occur differently in time and are strongly coupled to the specific phone identities. As explained in the previous section, vowel-fricative transitions occur rather quickly and a transition involving a burst (such as a plosive) is abrupt. By contrast, the vowel-nasal class behaves far more steadily. All these transitional time differences are referred to as the dynamics of change characteristics for phone transitions. The acoustic contextual influence can, however, be seen as purely an effect caused by two phone identities interacting as a transition takes place. In some cases

(such as nasals) this effect is so strong that it influences the frames throughout a whole segment. In addition, these two effects are also seen to be superimposed on each other for different transitions.

We continue analysing the observed co-articulation in the current section. Specifically, the trajectory functions allow investigation of the following additional parameters:

- identification rate of phone transition boundaries,
- the goodness of fit per trajectory,
- the difference between monophone unit estimates, and
- the trajectory slope at the phone transition boundary.

We analysed these measurements over all test data, and for specific phone classes. We also report the standard deviation of these measures as an indication of intra-class variability. Firstly, the technique used relied on the phone transition boundaries that are determined by an ASR system. Did the contextual difference that was measured accurately predict these boundaries? Section 3.4.1 investigates to what extent this assumption holds. Since co-articulation effects manifest differently for different phone contexts, we conducted experiments that considered broad phone transitional classes (see Section 3.4.2).

3.4.1 Boundary tracking

A transition boundary is identified by the tracked trajectories only if the polynomial functions (describing the co-articulation effect for the diphone transition) cross. If this occurs, one can evaluate how close these transition boundaries are from the version obtained from the HMM-based ASR system. Only some phone transitions produce pairs of trajectories that cross each other; Table 3.3 lists the number of phone transitions that can be identified by using polynomial function crossing points.

The technique described in this chapter restricted us to an analysis of the subset of phone transitions where the trajectories crossed. The difference measures (introduced in Section 3.3.2) combined the elements of the feature vectors, which may not always allow phone transition tracking. It was not clear exactly why the trajectories of some phone transitions did not cross and consequently it was not clear whether this method of trajectory tracking could be used to describe such phone transitions. The number of trajectories that did cross was sufficient to continue with the analysis in Table 3.3. Therefore, in Table 3.5 we continued to evaluate these transitions only. Since this was

Measure (order)	**# Usable boundaries**	**% Usable boundaries**	$SE_{segment}$	**Diff (# frames)**
Euclidean (1)	42 552	80.9	6.345	1.828
Euclidean (2)	47 909	91.1	4.318	2.100
Euclidean (3)	48 737	92.7	2.990	1.897
Euclidean (4)	49 312	93.8	2.311	1.846
Correlation (1)	41 502	78.9	0.964	1.839
Correlation (2)	46 430	88.3	0.569	2.103
Correlation (3)	47 534	90.4	0.339	1.933
Correlation (4)	48 101	91.5	0.240	1.871
Dot product (1)	39 526	75.2	44.471	1.959
Dot product (2)	46 079	87.6	25.051	2.314
Dot product (3)	46 688	88.8	13.976	2.039
Dot product (4)	47 272	89.9	9.580	1.971

TABLE 3.3: *The number of tracked trajectories that cross (usable boundaries), analysed in terms of the average distance in frames (Diff frames) from the known transition boundaries and the goodness-of-fit ($SE_{segment}$) for different orders of polynomial functions. For all measures, the shape of the second-order function introduces additional error and the third- or fourth-order functions performed much better.*

a preliminary analysis, later we refined the model (Chapter 5) to track feature channels individually and in the analysis in Chapter 7, we investigated all diphone transitions more closely.

Given the total number of transitions in the test data (52 571), we noted that the Euclidean distance identified more transition boundaries than the correlation or dot product measurements. This finding agrees with the result in Section 3.3.2. In fact, 92.7% and 93.8% of segments in the data for the third and fourth order polynomials do have trajectories that cross.

We calculated the average distance (in frames) for these usable boundaries between the identified and known phone transition boundaries. This gave a clear indication of the boundary-tracking capability of the trajectory functions. (Note that the ASR-based boundaries are also estimates rather than an indication of a basic truth.) In all cases, it is clear that the boundary-tracking capability of a second-order polynomial is limited for these values. The shape of the second-order function introduces an additional error. A first-, third- or fourth-order function performs much better.

Table 3.3 lists the goodness-of-fit ($SE_{segment}$) for the different polynomial functions, calculated by taking the average of the mean square error values that describe the fit of the two individual polynomial functions. Since the higher-order functions were used to estimate trajectories, a closer fit was obtained and the mean square error decreased. Because this might lead to overfitting, we selected a third-order polynomial for the

Transition group	**Slope 1**	**Slope 2**	**Diff reference values**	σ_1	σ_2
vowel-fricative	0.426	−0.236	3.064	0.510	0.516
vowel-stop	0.427	−0.203	2.925	0.603	0.592
vowel-affricate	0.353	−0.205	2.964	0.432	0.398
vowel-nasal	0.376	−0.164	2.493	0.639	0.635
vowel-semivowel	0.230	−0.180	2.413	0.490	0.515
vowel-vowel	0.168	−0.182	2.561	0.292	0.289
vowel-liquid	0.164	−0.175	2.551	0.349	0.388
vowel-aspirate	0.123	−0.063	2.122	0.440	0.466
nasal-liquid	0.347	−0.290	2.782	0.495	0.441
nasal-fricative	0.295	−0.317	2.819	0.531	0.485
nasal-affricate	0.311	−0.289	2.486	0.303	0.313
nasal-semivowel	0.566	0.001	2.108	1.311	0.924
nasal-stop	0.254	−0.266	1.958	1.030	0.943
nasal-aspirate	0.230	0.008	1.792	0.690	0.896
nasal-nasal	−0.226	−0.394	0.983	1.553	1.746
liquid-fricative	0.499	−0.315	3.084	0.528	0.503
liquid-affricate	0.358	−0.341	3.308	0.642	0.482
liquid-stop	0.371	−0.240	2.898	0.715	0.665
liquid-liquid	0.231	−0.207	2.944	0.269	0.298
liquid-aspirate	0.102	−0.126	2.440	0.436	0.415
liquid-semivowel	0.074	−0.084	3.030	0.241	0.262
fricative-semivowel	0.259	−0.323	3.140	0.429	0.457
fricative-stop	0.142	−0.154	1.760	0.427	0.444
fricative-aspirate	0.253	0.073	2.037	0.424	0.596
fricative-fricative	0.057	−0.068	1.450	0.539	0.490
fricative-affricate	0.088	−0.028	1.510	0.195	0.211
stop-semivowel	0.006	−0.493	2.573	1.403	1.215
stop-affricate	0.185	−0.107	1.630	0.332	0.308
stop-aspirate	0.059	0.260	1.583	1.080	1.300
stop-stop	0.119	−0.051	1.063	0.700	0.573
semivowel-affricate	−0.063	−0.510	2.951	0.556	0.320
semivowel-aspirate	0.321	0.0156	2.119	0.507	0.466
affricate-aspirate	0.503	0.096	1.635	0.301	0.362

TABLE 3.4: *Slopes, Euclidean distance to the monophone means and the standard deviation of the slopes for third-order polynomial functions at ASR diphone transition boundary. (Ranked according to the steepness of the slopes for every broad transitional phone class.) Phone transitions with steep slopes also yield good separation for the mean difference between unit estimates.*

remainder of this analysis: the shape of a third-order polynomial lends itself well to describing the behaviour of a trajectory near and crossing a phone boundary. These polynomials allowed us to focus on the co-articulation emanating from a single phone transition.

3.4.2 Effect of broad phonemic classes

Transition group	**# Correct classifications**	**% Accuracy**
vowel-affricate	640	95.8
vowel-fricative	8 268	91.6
vowel-semivowel	2 096	83.1
vowel-stop	9 142	79.8
vowel-nasal	5 005	76.5
vowel-vowel	1 143	70.2
vowel-aspirate	693	65.4
vowel-liquid	4 790	62.9
nasal-affricate	93	92.1
nasal-fricative	862	84.4
nasal-semivowel	125	79.6
nasal-aspirate	38	64.4
nasal-stop	1 040	63.1
nasal-liquid	183	60.0
nasal-nasal	23	29.1
liquid-affricate	53	100.0
liquid-fricative	709	94.8
liquid-stop	1 969	83.9
liquid-semivowel	230	79.6
liquid-liquid	44	73.3
liquid-aspirate	30	65.2
fricative-semivowel	353	94.9
fricative-aspirate	70	75.3
fricative-stop	1 864	69.5
fricative-fricative	270	59.3
fricative-affricate	34	57.6
stop-semivowel	408	72.6
stop-affricate	77	63.6
stop-aspirate	52	45.6
stop-stop	227	35.0
semivowel-affricate	9	81.8
semivowel-aspirate	15	68.2
affricate-aspirate	3	100.0
total	40 558	77.1

TABLE 3.5: *Number of correct classifications using mean frame-based values and known ASR boundaries for specific transitions (only calculated for a subset where polynomials intersected).*

Different classes of phone transitions reveal interesting trajectory effects. Specifically, we evaluated five parameters to categorise the trajectories formed for different classes, namely:

1. The slope of the polynomial function trajectory for the first unit estimate variable,

2. The slope of the polynomial function trajectory for the second unit estimate variable,

3. The Euclidean distance of the monophone means (the difference between the two unit estimate values),

4. The standard deviation σ_1 of the first slope, and

5. The standard deviation σ_2 of the second slope.

Table 3.4 shows the above values for the different phone transition classes constructed according to the CMU phone groupings [85]. By ordering according to the steepness of the slopes, we found that phone transitions with steep slopes also yielded good separation for the mean difference between the unit estimates. In fact, we calculated the average of the difference values for the vowel-fricative, vowel-affricate, nasal-fricative and nasal-affricates to yield correct phone identity separation percentages of 91.6, 95.8, 84.4 and 92.1 respectively (Table 3.5). Similarly, the nasal-nasal class had a low separation of average difference values (0.194 – see Table 3.4) for the few phone transitions (29.1%) that did yield correct phone identity separation. There were exceptions to the rule. The nasal-liquid class had good separability and steep slopes but the identity detection based on the average difference values was at 60.0%. In general, similar classes (such as nasal-nasal or fricative-fricative) have the weakest separability, as could be expected.

We observed that the standard deviations of the slopes, σ_1 and σ_2 were similar in magnitude for particular phone classes, indicating a similar variability in the intra-class diphone transitions. Interestingly, the magnitude of the two slopes was typically not equal, with the divergence from the first mean occurring more quickly than the approach towards the second mean.

3.5 Conclusion

The initial experiment discussed in this chapter showed the presence of strong contextual effects in speech data. For further analysis, we refined the contextual modelling approach as explained in Chapter 5. The complexity of co-articulatory behaviour requires high-quality speech data covering many examples of similar contexts. The next chapter describes how the speech data used in later experiments was created.

Chapter 4

Corpus design for trajectory modelling

4.1 Introduction

The analysis in Chapter 3 gives some indication of the nature of the co-articulation effects for MFCC features. Moreover, the tools used enabled analysis on a per-frame basis. The general impressions gained from these investigations indicate several complexities:

- The acoustic contextual influence of the monophones to the left and right of a diphone transition seem to vary greatly.
- The speed of the transition also seems to play an important role.
- The above two effects combine to produce an even larger number of different phone transitions.
- Co-articulatory trajectories (for the same transitional context) vary among speakers.
- Multiple examples of the same transition vary in the same manner (but to a lesser extent) for the same speaker.

In view of the observed intricacies, we constructed a special high-quality speech corpus, of a single speaker, so that we could specifically analyse co-articulation. Section 4.2 describes how the ATTC was created. Automatic quality measures were used for selecting high-quality utterances, for which specific training data sets and test data sets were created (Section 4.3). Finally, we give details of the development of an ASR system using this corpus, and also the recognition accuracy obtained (Section 4.4).

4.2 Corpus construction

The process of corpus construction entailed the collection of about 6000 short utterances, providing a large corpus of speech of a single male speaker. By considering a single speaker only, it was possible to focus on contextual effects first, without inter-speaker differences complicating the results. Recordings were made for the ATTC by using a list of short Afrikaans prompts (one to five words in length) with balanced phonetic coverage [86]. The use of balanced prompt lists ensured sufficient contextual variation. To describe the contexts, the same Afrikaans phoneset as in [87] was used. Throughout the remaining chapters of this thesis we employed a standard computer-readable notation for phone descriptions ("Speech Assessment Methods Phonetic Alphabet") [88].

We used the smartphone-based tool *Woefzela* [89] for collecting the speech data. With this software, a recording session typically consisted of 500 prompts. About 11 sessions of between 400 and 600 prompts each were recorded over a four-month period. An experienced respondent was chosen and the recordings were made in a very specific manner; background noise was minimised and the respondent mostly succeeded in ignoring prompts containing proper names, spelling errors and English words.

Quality control was performed, and scores were estimated given a phone-based dynamic programming (PDP) algorithm [90]. Such verification of the speaker's pronunciations resulted in the selection of a high-quality (aligned) set of recordings [91]. For this purpose, acoustic models, forced alignment and phone recognition (using a flat phone grammar) were obtained on all the data. The insertion penalty value for the free recognition was kept at a value of -15. After this procedure, the number of utterances showing perfect alignment (a PDP score of 0) totalled 4 974. From manual inspection, it was clear that lower PDP scores were caused by recording, dictionary and speaker error, including minor mispronunciations (for example unrounded vowels, where rounded vowels were required). These were therefore removed from the data, in order not to introduce unnecessary variability during detailed trajectory analysis. This process may have excluded some variability that is normally acceptable, but it is uncertain to what extent. For our purpose we decided first to work with the data showing perfect alignment. At a total duration of approximately 3 hours and 20 minutes this corpus produced a higher triphone coverage for a single speaker than is typically available from general ASR corpora.

4.3 Experimental data sets

The master training and test data sets were selected from the "clean" data set. All the diphone transitions that occurred 30 or more times in the clean data set were retained,

and greedy selection was used for selecting the test utterances until the test set contained at least three examples of each of these diphones. The remaining clean utterances formed the training data set. After completing these steps, the number of utterances selected totalled 902 and 4 072 for the test and training data sets, respectively.

Transitions	**All data**	**Train**	**Test**
Total number	783	769	678
30 examples	470	436	173
< 30 examples	313	328	505
< 3 examples	85	86	127

TABLE 4.1: *Number of unique diphone transition labels in the data for various selection stages of the master training data set and test data sets. After selection, 86.7% of these diphones occurred in the test data, while 87.2% of diphone transitions had three or more examples in the training data.*

Table 4.1 displays the total number of unique transition labels given to show that there were more than 30 examples for many labels (470). A representative test set could be generated, given that 678 (86.7%) of the total number of diphone transitions were seen and 551 of these, 70.4%, were seen three or more times. Moreover, the master test set was only about $\frac{1}{5}$ the size of the master training data set, where 769 (98.2%) of diphone transitions were seen and 683 (87.2%) had three or more examples.

In addition, a development data set (for parameter selection) was required. This set of 873 utterances was selected from the 4 072 training utterances, using the same strategy as that for the initial training and test set partitioning. This time, all the diphone transitions that occurred 30 or more times in the training data set were retained, and development utterances were then greedily selected until the development data set contained at least three examples of each of those diphones. When using the development set for evaluation purposes, the training data set therefore consisted of fewer utterances (a total of 3 199).

4.4 Recognition accuracy

We used a standard HMM-based ASR system trained on the 4 072 clean recordings of the master training set to perform flat phone recognition. A context-dependent cross-word phone recogniser with tied-triphone models was employed; 39 MFCC features were used, including the first 13 and their first- and second-order derivatives. These features were computed with a window size of 25 ms and a frame rate of 10 ms. Semi-tied transforms were applied. Each triphone model has three emitting states with seven Gaussian mixtures per state and a diagonal covariance matrix. We calculated phone recognition accuracy by matching reference and hypothesis phone labels of recognised

utterances using dynamic programming (DP) [92]. Subtracting the number of insertions from the number of correctly detected labels and normalising this score to the total number of labels of the hypothesis provides a base score. We obtained the reported recognition accuracy values, multiplying this base score by 100, yielding a percentage.

Verifying the accuracy of phone recognition on the master test set, using a flat-phone grammar and a phone insertion penalty of -20, yielded remarkably high phone accuracy. A value of 93.72% could be obtained. A confusion matrix of the test phone accuracies revealed in particular that four phone labels had a correctness of less than 83.00% and that two of these labels performed below a correctness of 42.00%. Table 4.2 lists these phones and the specific phone labels with which they were most frequently confused. Manual verification of a few samples confirmed that for this speaker, the /9/ and /2:/ phones indeed sounded very similar to the /@/ and /i@/ phones, especially for utterances made at a higher speaking rate. In fact, this derounding effect of the first three vowels, indicated in Table 4.2, Wissing [93] described in his research as an important difference between Afrikaans and other Germanic languages.

Phone label	**Correctness**	**Mainly confused with**
9	41.20	@
2:	41.90	i@
y	70.00	i
@u	83.00	@, @i, u

TABLE 4.2: *Correctness of and alternatives for confusable phone labels, clearly displaying the derounding effect in Afrikaans.*

4.5 Conclusion

This chapter introduces the ATTC that was analysed during all further experiments on the detailed effects of co-articulation. The ATTC contains sufficient examples of context-dependent phones with adequate quality. Chapter 5 explains the more precise modelling of co-articulation using a trajectory tracking model.

Chapter 5

Trajectory tracking model

5.1 Introduction

A phone transition-based approach was taken to the initial analysis of co-articulation effects (Chapter 3), specifically grouping the closely related frame sequences. These sets of frames could be used for extracting characteristic trajectories when modelling the acoustic change during a particular diphone transition: a convenient segment of speech to model a single change from one acoustic unit to the next, which started and ended at the more stable centres of acoustic units. When analysing the speech segments, experiments showed that there was a broad variety of frame-based dependencies for diphone transitions.

Since these effects were ultimately encoded in the raw features, another more direct approach would be to model the feature trajectories of a segment for each feature channel individually. With typical ASR features such as MFCCs, this would mean that 13 frame-based trajectories could be extracted to describe the acoustic change of a single diphone transition (segment). (We chose to continue using diphone segments, since the rate of acoustic change was expected to be fastest near the ASR boundary between adjacent phones and lowest at phone centres. Plots of the 13 MFCC vectors support this idea, as the magnitudes of individual feature channels are relatively stable near phone centres.) Probing the MFCCs of multiple segments revealed definite changes near the ASR boundaries.

Section 5.3 explains how (in this chapter) we set out to track the cepstral trajectories of individual segments, using a simple piecewise linear model. By taking this approach, we attempted to divide the frames of each segment further into three sub-segments, and found that the central sub-segment modelled most change. To evaluate how well these

simple models could represent cepstral frames, a mean square error (MSE) measure was defined, estimating model mismatch for specific transitional classes.

Section 5.4 describes the additions made to the standard segment model. Multiple segments were effectively joined together to model the transitions of a whole utterance. Secondly, not all segment-based transitions of MFCC features displayed a single change. More line pieces could be used to represent the acoustic change in these transitions better.

Finally, the tracked trajectories were investigated. We conducted a set of experiments to verify the trajectory behaviour of MFCC features (Section 5.5). Similar transition examples were grouped together and the consistency of characteristic parameters was evaluated. We describe the transition parameters of separate model parts to measure to what extent the resulting trajectories are systematic. Given the training data, the predictors of these model parts could be trained and we used the predictions to judge the importance of specific contexts for transition models.

5.2 Terminology

Table 5.1 provides the definitions of a number of additional terms defined in this chapter and used in subsequent sections.

Term	Interpretation
Feature value	A single number representing the value of a single feature dimension at a particular time instant.
Feature channel	A single dimension over a given time period.
Trajectory model line pieces	Linear parts of simple piecewise trajectory models.
Trajectory model	The linear representation of phone transitions.
Stable values	Start-and-end trajectory model line pieces for a single segment.
Change descriptor	Central line segments of trajectory model for a single segment.
Model alignments	Time indices where the stable value and change descriptor line pieces connect for a single segment.
Transition model	Trajectory models where each segment is modelled individually.
Utterance model	Trajectory models where segments are connected though the sharing of stable values.

TABLE 5.1: *Terminology used throughout this work.*

5.3 Piecewise linear models

As mentioned in Section 5.1, the definite transition at the cepstral level lends itself to piecewise linear modelling. This section first defines a transition model. The different co-articulatory mechanisms require such models to be capable of tracking change segments

of variable length at the sub-phone level. It is also clear that co-articulation effects are not constrained to the frame level. Appropriate models need to operate at the segment level and even longer-term effects must be considered. By searching for variable-length positions for these segments one could characterise the detailed transitional behaviour and obtain a direct comparison between the modelled trajectories and the actual speech data. Different modelling choices may be compared by measuring the consistency of the tracked changes, leading to new insights into cepstral transition behaviour.

5.3.1 Transition model

We used piecewise linear approximation to model the transitional behaviour. Three line pieces were used to fit the cepstral values of a single MFCC feature channel (cepstral transition), using least-squares optimisation. Figure 5.1 depicts this modelling strategy. Similar to the analysis in Chapter 3, we modelled a phone transition as a diphone, using only the 50% of monophone frames closest to the ASR boundary. We restricted the start-and-end line segments to constant values (linear with zero slope), and modelled the transition between these two values with a straight line of variable slope. Furthermore, the constant line segments (the start-and-end line pieces) were required to be associated with at least θ frames serving as trajectory anchor points, with $\theta = 1$ in the current work. These constant line segments were called *stable values* and the remaining central line segment was called the *change descriptor*.

The estimation of the centre line piece was not explicitly associated with any data points; instead, we utilised the zero order anchor points (stable values) and drew the first order line between the end and starting indices of the two anchor points. We searched for these indices (model alignments) by optimising the squared error SE across all three line segments. This also yielded a single error value for the specific approximation. Optimising the squared error enabled us to find the best model alignments. In order to compare the different options, the squared errors (SE_{frame}) of each stable value and change descriptor at each instant were estimated as:

$$SE_{frame} \quad = \quad SE(p, \hat{p}, \omega) \text{ where } \omega = (p_f, \hat{p}_f) \tag{5.1}$$

where for the frame f, $\hat{p}$ is the trajectory function of feature channel c and p_f the true feature value. Then $(p_f - \hat{p}_f)^2$ is the squared residual. This is followed by the mean square error (MSE_{trans}) across features:

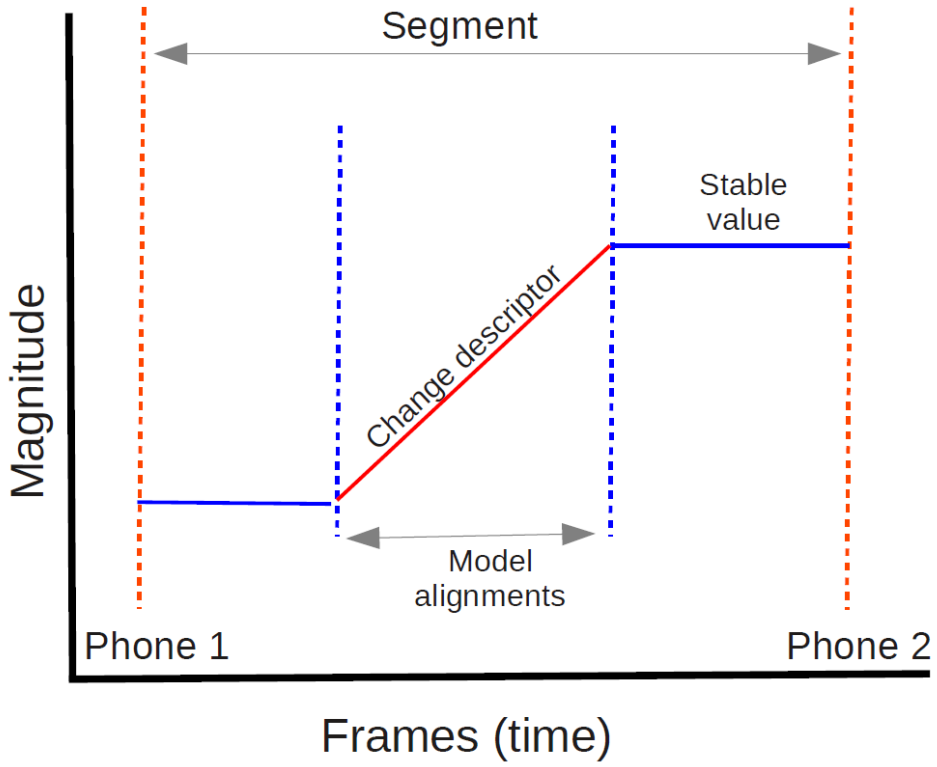

FIGURE 5.1: *Depiction of the transition model.*

$$MSE_{trans}(c,d) \quad = \quad MSE(p,\hat{p},\omega_{cd}) \text{ where } \omega_{cd} = \bigcup_{s=1}^{S_d} \bigcup_{f=1}^{F_s} (p_{fc},\hat{p}_{fc}) \tag{5.2}$$

for the feature channel c and this segment of a diphone transition d. F denotes the total number of frames for the segment. The measure is calculated across all diphone samples S_d and segment frames F_s.

Once optimised, this model provided the following values (Figure 5.2):

S1	parameter value at initial stable value
$T1$	frame at start of the transition
T_{mid}	centre of the transition
$T2$	frame at end of the transition
S2	parameter value of final stable value.

(5.3)

As these were calculated for each feature channel individually and could be calculated over every single transition individually, the resulting set of parameters could be rather

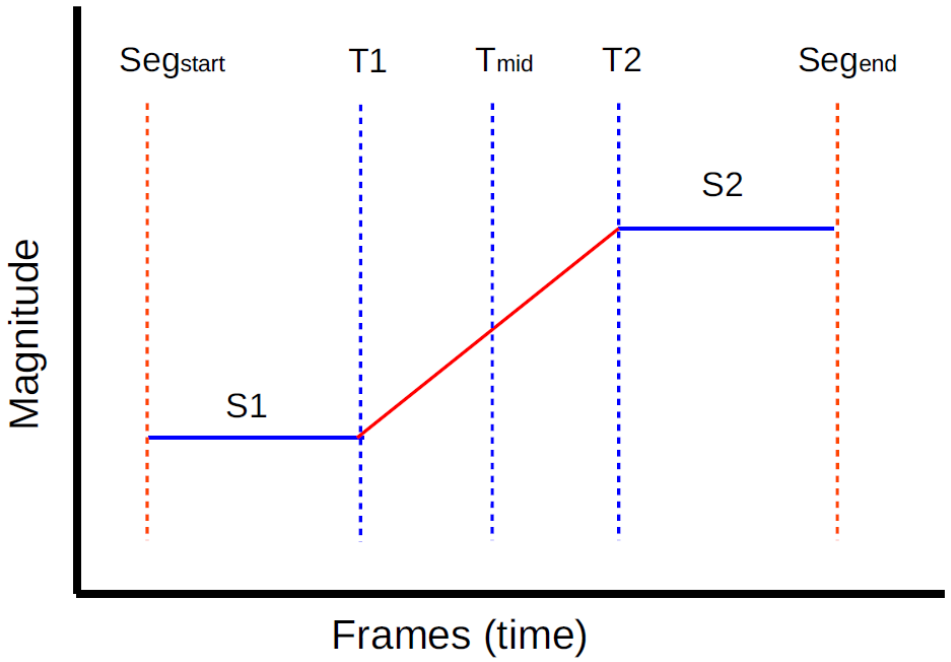

FIGURE 5.2: *Characteristic representation for a single transition.*

large. (Since each of these parameters was an independent measurement, we denoted them as separate scalar values.) Seg_{start} and Seg_{end} denote the start and ending frames of the diphone segment respectively.

5.3.2 Model fit

Figure 5.3 displays the linear approximations for the first four cepstra of a single diphone transition example. The separate model parts of the first cepstral coefficient (MFC 1) can be clearly identified: two stable values (frames 9 – 15 and frames 17 – 21) and a single change descriptor (frame 15 – 17, connecting the stable values). As this is a transition model, the stable values are anchored to the start and ending frames of the diphone segment (frames 9 and 21 respectively) and do not extend to adjacent transition frame numbers (1 – 8 or 22 – 26). For all cepstra, a single definite transition was observed near the ASR boundary.

By means of Equation 5.2, the MSE measurement can be calculated for the separate model parts, or for the whole piecewise approximation of the specific feature channel and the multiple transition examples, by including the relevant frames. To measure how well the different trajectory estimation approaches compare with the actual observed MFCC feature vectors, the MSE measurement ($MSE_{diphone}$) of trajectories is particularly useful. This value allows direct comparison of phone transitions, with regard to

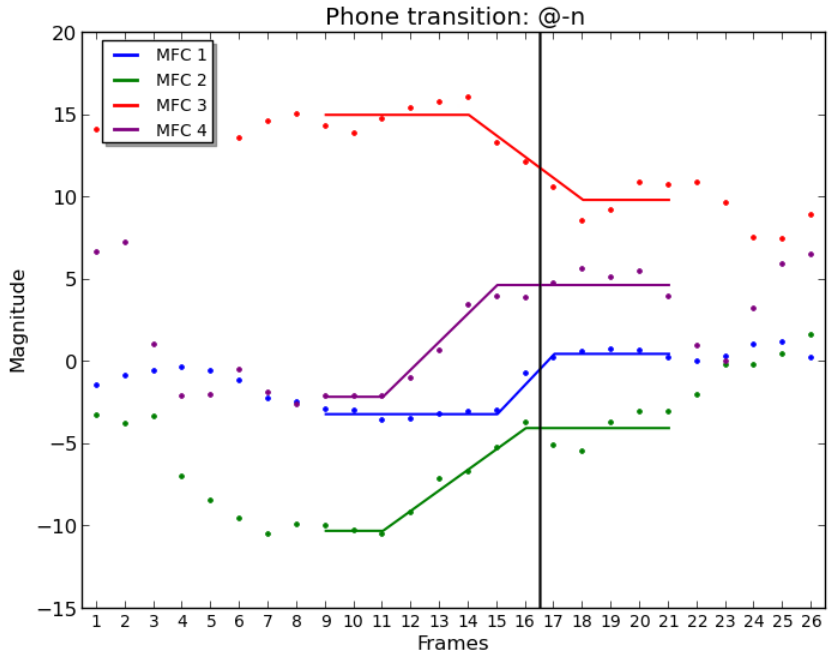

FIGURE 5.3: *Piecewise linear model fit of the first four cepstra of the diphone transition /@-n/ using 3-piece transition models.*

the training data, across all cepstral transition models. The $MSE_{diphone}$ measurement can be calculated similarly to Equation 5.2:

$$MSE_{diphone}(d) \quad = \quad MSE(p, \hat{p}, \omega_d) \text{ where } \omega_d = \bigcup_{s=1}^{S_d} \bigcup_{c=1}^{C} \bigcup_{f=1}^{F_{cs}} (p_{fc}, \hat{p}_{fc}) \tag{5.4}$$

This is the MSE for a particular diphone d, over the set (ω) consisting of all frames of all samples S_d that contain this diphone, summed over all feature channels C. Every transition generates F_s squared errors (one for every frame) and there are $C = 13$ of these feature channels (one for every MFCC coefficient). Finally, to represent the entire set of transitions with a single error value, the summation of the contributions from each class is evaluated:

$$GMSE_{diphone} \quad = \quad \frac{1}{D} \sum_{d=1}^{D} MSE_{diphone}(d) \tag{5.5}$$

where $GMSE_{diphone}$ is the mean trajectory MSE estimated for S examples of a contextual class and a total of D classes.

The global average $GMSE_{diphone}$ is taken specifically across the $MSE_{diphone}$ values, since this allows the MSE contribution of each transition label to be weighed equally. Single MSE values can also be calculated for all frames and transitions, but this might skew the measurement toward more frequent labels.

5.4 Transition model improvement

The first priority in modelling phone transitions was to represent speech data accurately, so that systematic effects (if present) could be identified. Although the transition model description given in the previous section can describe many key changes for typical feature channels, not all the variations are captured with the required precision. Specifically, the feature values may include characteristic double transitions for certain channels. Moreover, the fact that a single transition forms part of an entire utterance introduces significant error. The next sections therefore extend the linear model definition to reduce this error with improved trajectory representations.

5.4.1 Connecting segments

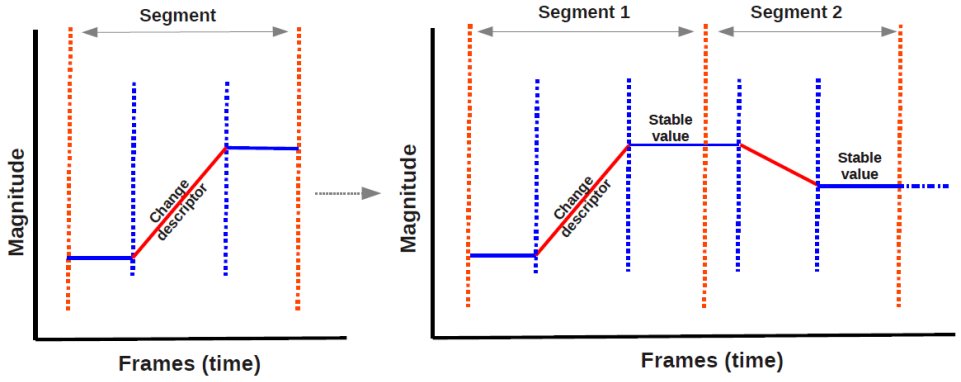

FIGURE 5.4: *Depiction of an utterance model connecting segments by forcing stable value overlap.*

In the standard model, each segment is modelled separately. This means that stable value estimates of two adjacent models may not necessarily be the same, but exhibit a "gap". The piecewise linear modelling algorithm must be extended to represent the more stable parts of phone transitions (segments) correctly. We adjusted the piecewise linear approximation algorithm to model the entire utterance in a single process, eliminating this artificial gap by forcing the adjacent stable values to be equal to one another. The new configuration is depicted in Figure 5.4. Finding the utterance model for a feature channel was accomplished as follows:

- Estimate transition models for all the transitions in the utterance.
- Copy the model alignments for each phone transition.
- Fit the trajectory model line pieces, connecting the model alignments, for each feature channel of the utterance.

In this two-step estimation process, we first estimated a complete set of transition models for all segments of a whole utterance. The construction of a completely new utterance level trajectory model (one for every feature channel) was performed in an additional step. We directly applied the model alignments of the transition model set on a per-segment basis. From left to right, we fitted a change descriptor and an ending stable value (except for the first transition and for all feature values up to the time index where the next change descriptor connects), re-using the last stable value of the previous segment as the first stable value of the current segment.

5.4.2 4-piece segments

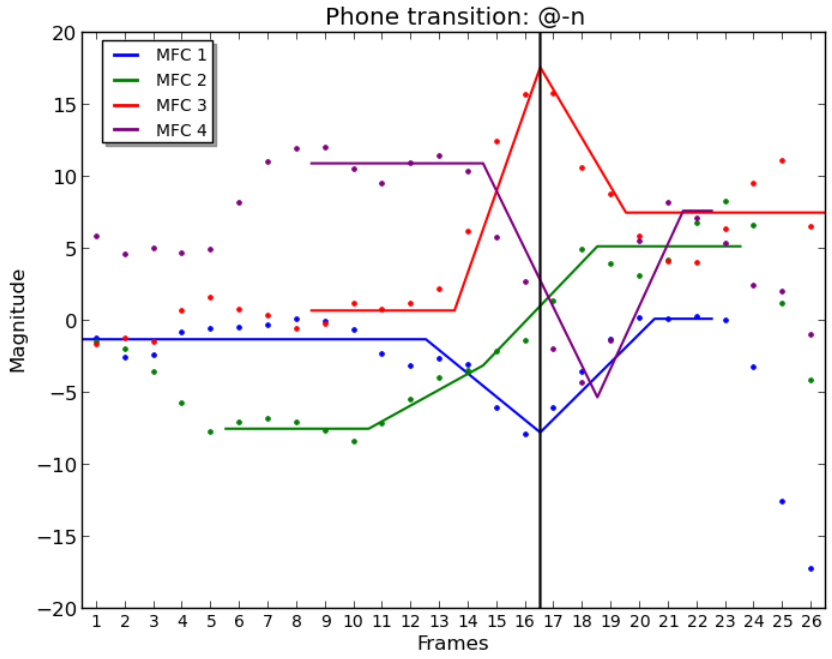

FIGURE 5.5: *Piecewise linear model fit of the first four cepstra of the diphone transition /@-n/ using 4-piece utterance models.*

Figure 5.5 shows another example of the same diphone transition as the example in Figure 5.3, but in a different context. Though some of the cepstral transitions can be seen to be moving in (relatively) similar directions and some start-and-end positions

seem to agree, it is clear that the transition itself behaves rather differently. Instead of single transitional changes, characteristic peaks and troughs are now formed. This behaviour is seen quite frequently for certain transitions and feature channels. It is possible that in such cases a more elaborate change descriptor could be of value to model the change accurately.

To improve the change descriptor representation, we implemented a 4-piece symmetrically constrained model (Figure 5.6). This allowed the change descriptor to have a freely varying centre-point (connecting the two change descriptor line pieces). As a final constraint, the change descriptor had to be symmetrical along the time axis during model alignment. Using this configuration, the change descriptors consisted of two line pieces, which were kept at an equal length. This requirement drastically reduced the search space to find model alignment, compared to the requirement when any of the four line pieces could be of an arbitrary length, and might be more robust in detecting specifically the peaks and troughs. The final model fits (when connecting segments) of the utterance might, however, find "shared" stable values, which differ from those used during model alignment, leading to non-symmetric change descriptors (Figure 5.6).

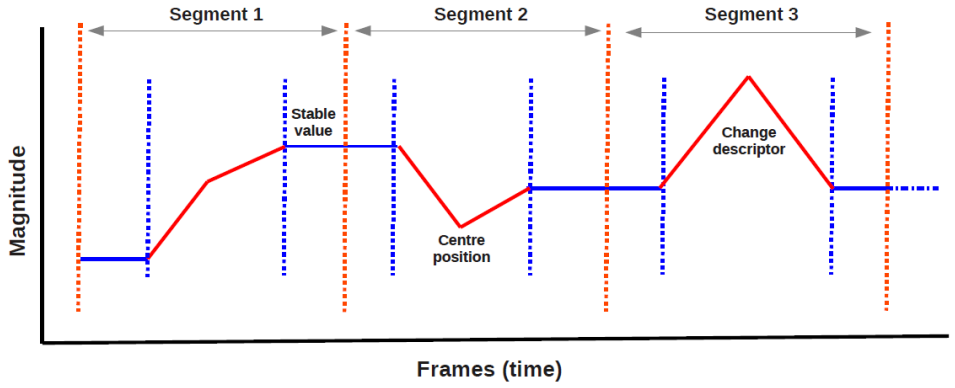

FIGURE 5.6: *Depiction of 4-piece utterance model segments.*

Figure 5.5 depicts the estimations of linear approximations with 4-pieces for the first four cepstra. Since this is also an utterance model, the stable value parts are now shared with the adjacent transition models (e.g. the stable value of the first cepstral coefficient, MFC 1, is now fitted to frames 1 – 12 where this diphone transition example only begins at frame 9).

5.5 Approximation sufficiency of trajectories

Given a set of trajectory models and adequate model tracking, the next key goal was to define the measures (based on model parameters) that could identify systematic trajectory characteristics. This section defines the measures that evaluate the consistency

of stable values and of change descriptors. More consistent parameters are a more favourable choice for the representation of the transition model and may indicate increased systematic trajectories. Various other values could be calculated from the main parameters listed in Section 5.3.1. Examples include the slope (the gradient of the first order line) of the transition, the duration of the change descriptor ($T2 - T1$) and the size of the transition (difference between the stable values, $S2 - S1$).

Once adequate representation of the training data has been achieved, these parameters can be used to characterise specific transitions. We evaluated the consistency of a measurement across multiple samples of the same transition in the data set. Grouping together multiple examples of similar transitions provided a way of measuring systematic behaviour. We found that the technique of employing stable value predictors was particularly useful to compare the effect of grouped trajectories, because a particular test trajectory could be predicted on the basis of the trained, stable value predictors.

5.5.1 Model consistency

Sections 5.5.1.1 and 5.5.1.2 describe the two consistency measurements were used for identifying systematic behaviour of cepstral transition trajectories. Different modelling options could then be compared directly (for the same transition examples).

5.5.1.1 Reference stable values

Reference stable values were estimated by using the training data set. Once an initial set of trajectories had been fitted to the training data, the mean was estimated for the stable (constant) parts of every particular context required. After estimating the reference stable values, these values could be predicted for unseen samples of the test set. We evaluated the model fit (as described in Section 5.3.2) in order to compare the trajectories obtained with predicted stable values.

Trajectory models with fixed reference values behave differently from the model alignment algorithm than free trajectory models do. To investigate this, two estimation options were compared:

- Trajectory estimation with fixed reference values, and
- Trajectory estimation with fixed reference values and constrained alignment.

Since model alignment estimation is dependent on the chosen stable values, using predicted stable values instead of the actual values leads to additional error. An intermediate option would be first to fit the free trajectories (to find transitions), then to enforce stable values (from predicted reference values). This combined information would then constitute the final trajectory.

Moreover, stable values have to be shared between adjacent segments for utterance models. To meet this requirement, when fixed reference stable values were required, we fitted the mean of the two reference stable values contributing to a single shared stable value. In this way, segments were connected to form a single trajectory for the whole utterance.

5.5.1.2 Change descriptor consistency

We evaluated change descriptor behaviour in terms of temporal information and defined two representative parameters to evaluate consistency: (1) relative position to ASR boundary and (2) absolute duration.

In the speech segmentation process, a single ASR boundary was obtained for every phone transition. This boundary had the same location for all 13 cepstral channels and was useful to provide an initial alignment for other similar transition examples. In this way, we compensated for the fact that not all examples were of equal length. Measuring the centre positions (exactly half-way between the model boundaries) of the change descriptors and relative to the ASR boundary provided a good indication of the position where most of the change for a cepstral transition occurred. We also used the absolute duration, which was the length of the change descriptor as defined by the model alignments, to complete the measurement description. The position and duration measurements were both given in terms of frame units.

For each cepstral transition class, a parameter p, a channel c, a specific diphone d and over S samples, we estimated the sample standard deviation $\hat{\sigma}_{trans}$, namely:

$$\hat{\sigma}_{trans}(c,d) \quad = \quad \sqrt{\frac{1}{S}\sum_{s=1}^{S}(p_s - \hat{\mu}(p_s,\omega))^2} \text{ where } \omega = \bigcup_{s=1}^{S_{cd}}(p_s) \tag{5.6}$$

and

$$\hat{\mu}(p_s,\omega) \quad = \quad \frac{1}{S}\sum_{s=1}^{S} p_s \tag{5.7}$$

where p_s is the measured parameter value for sample segment s and feature channel c, and found $\hat{\mu}(p_s, \omega)$ the mean and $\hat{\sigma}(p_s, \omega)$ the standard deviation for a total of S example segments of the cepstral channel c. To represent an entire set of cepstral transitions with a single consistency value ($G\hat{\sigma}_{trans}$), we summed the contributions from each cepstral transition class ($\hat{\sigma}_{trans}$):

$$G\hat{\sigma}_{trans} = \frac{1}{CD}\sum_{c=1}^{C}\sum_{d=1}^{D}\hat{\sigma}_{trans}(c,d) \tag{5.8}$$

A total of R standard deviations (where $R = CD$; for C feature channels and D phone transition labels) were summed in this way.

5.5.2 Features for transition modelling

The experiments described in this chapter were performed on phone transitions that had been selected to ensure that data scarcity would not interfere with the investigation. This section contains a discussion of the selection process. Each phone transition was selected from the high-quality set of speech recordings (of a single speaker). For this purpose, we used the ATTC with the dedicated training data and test data sets, as described in Section 4.2.

Any phone transition requires deriving the specific MFCC features and the appropriate speech segmentation. The specifics of the features used for modelling the cepstral transitions were kept exactly the same as those in Section 4.4 and the segmentation of the systems described there was used directly to determine the diphone segments. In addition (and after segmentation), though we still used a window size of 25 ms, the frame rate was adjusted (from 10 ms) to 5 ms. This yielded a better time resolution. Only the raw MFCC coefficient channels were used, not any of the derivatives. Finally, for every utterance, each of the MFCC vectors was associated with the phone-boundary alignments from above, which provided contextual labelling at the triphone level.

Given the test data and training data sets, a further selection process was used to select specific transition examples for the experiments conducted and described in this chapter. As described in Section 4.3, the total number of unique transition labels is given to show that there were more than 30 examples for a large number of labels (470). (All transitions with fewer examples were ignored for this transition model analysis.) After excluding the transitions that included the silence label, we made a final (per example) selection. A particular transition example was only allowed if the duration (in frames) was no more

than a single standard deviation from the mean. The result of this selection provided the 331 most frequent transition labels and transition examples that had low variability in speech rate.

5.5.3 Evaluating trajectories

The $GMSE_{diphone}$ measurement (as defined in Section 5.3.2) were used for comparison of various trajectory tracking techniques. These values were calculated for each model option. In the case of utterance models, we always converted to a valid transition model representation, ensuring that direct comparison of the phone transitions on a per-segment basis was valid. Model options with predicted stable value parts required a training data set and test data set to assess the trajectory tracking. Then the transitions of the test data set used the fixed stable value fits (applied during model estimation), as predicted by using the training data.

Model	**MSE train**	**MSE test**	**MSE stable values**	**MSE change descriptors**
3-piece (trans)	3.604 (1.265)	3.609 (1.306)	4.047 (1.448)	1.821 (0.794)
3-piece (utt)	7.710 (2.335)	7.692 (2.433)	8.299 (2.667)	5.415 (1.837)
4-piece (utt)	4.211 (1.243)	4.181 (1.242)	4.959 (1.481)	3.427 (1.106)

TABLE 5.2: *Overall $GMSE_{diphone}$ measurements for train and test data trajectories. Both mean and standard deviation (in brackets) values show the closely similar MSE values obtained with free trajectories for MSE train and MSE test and the cost of connecting segments.*

Table 5.2 lists the estimated $GMSE_{diphone}$ measurement values for free trajectories. Global MSEs were estimated for the phone transitions of both data sets (*MSE train* and *MSE test*) and two values given per measurement: the mean and standard deviation (in brackets) of the diphone transition class MSE, respectively. We observed that the error on the training set was in agreement with the test set results for free trajectories. Closely similar MSE values were obtained for *MSE train* and *MSE test*. There was a cost to connecting segments: overall, the error increased (as could be expected for the more constrained model). Separate model parts could also be evaluated, and the global MSE values for only the frames corresponding to the stable value (*MSE stable values*) or change descriptor parts (*MSE change descriptors*) of trajectory models are given.

Similar global MSE values could be derived for the test set (*MSE test*) when predicting fixed stable values. Table 5.3 gives a comparison of these values for the fixed stable value trajectories. To aid the comparison, a ratio was also determined between the global MSE values with every fixed stable value trajectory option and its corresponding free trajectory (*Free fit ratio*). The ratios between fixed stable value and free trajectories improve, although connecting segments increase the model error (as shown in the

Model	Fixed stable value	MSE test	Free fit ratio	MSE stable values	MSE change descriptors
3-piece (trans)	Monophone	24.553 (6.338)	6.803	27.636 (6.662)	8.861 (5.165)
	Biphone	19.228 (4.223)	5.328	21.908 (4.447)	6.756 (2.979)
	Triphone	19.118 (3.574)	5.297	21.961 (4.105)	6.974 (2.141)
3-piece (utt)	Monophone	24.715 (6.342)	3.213	27.775 (6.688)	11.437 (5.954)
	Biphone	21.655 (4.681)	2.815	24.531 (5.163)	9.658 (3.400)
	Triphone	19.399 (3.671)	2.522	22.238 (4.224)	8.610 (2.354)
4-piece (utt)	Monophone	12.829 (2.894)	3.068	19.050 (5.407)	9.132 (1.991)
	Biphone	11.325 (1.884)	2.709	16.796 (3.520)	7.966 (1.393)
	Triphone	10.585 (1.577)	2.532	15.586 (2.771)	7.570 (1.324)

TABLE 5.3: *Overall $GMSE_{diphone}$ measurements for predicted stable value (with different context size options) test data trajectories. The ratios (Free fit ratio) between fixed stable value and free trajectories improve, although connecting segments increase the model error.*

previous table). These ratios showed a decrease in error when predicting stable values (instead of estimating them for each phone occurrence). For utterance models, the error increases by a factor of less than three, compared to a value of six when using transition models.

As previously observed in [94], larger context sizes allow for more specific stable values and improved model fit. Consequently, we tested reference stable values of different context sizes (monophones, biphones and triphones) and found that the predicted stable value model fits improved up to the triphone context level (see Table 5.3).

5.5.4 Consistency of the change descriptor

We estimated the global consistency values $G\hat{\sigma}_{trans}$ of specific temporal parameters to compare the change descriptor behaviour. Once stable values had been estimated, the timing of the change descriptor of a specific transition was determined by finding the best fit from one stable value to another. This may not produce optimal model alignment, especially if a specific stable value does not suit a specific sample of a transition well. Table 5.4, demonstrates this by firstly estimating a global free trajectory baseline consistency $G\hat{\sigma}_{trans}$, for the transitions of the test set, of the parameter T_{mid}, indicated as *Cons (T_{mid})* in Table 5.4. Then the trajectories with and without (indicated as "No" under the *Fixed stable value* heading) fixed stable values were compared, where biphone and triphone context sizes were used.

Clearly, the free trajectory model alignments could be used for better change detection. We found that the measured change descriptor position was less consistent for trajectory models with fixed stable values (free trajectory models show most consistent positions for T_{mid} in all cases). When using models that had free trajectory alignments, the

Model	Fixed stable value	Cons (T_{mid})
3-piece (trans)	No	2.847 (1.215)
	Biphone	4.095 (1.753)
	Triphone	4.107 (1.753)
3-piece (utt)	No	2.848 (1.215)
	Biphone	3.924 (1.707)
	Triphone	4.024 (1.721)
4-piece (utt)	No	2.167 (0.751)
	Biphone	2.289 (0.832)
	Triphone	2.281 (0.828)

TABLE 5.4: *Overall consistency $G\hat{\sigma}_{trans}$ measurement of change descriptor position on test set. Free trajectories and 4-piece models provide better change detection.*

fixed stable values could still be applied without allowing the model to find new model alignments ($T1$ and $T2$ – see Section 5.5.1.1).

The more consistent change descriptor positions of the free trajectory models motivated further investigation of free trajectory alignments. To understand the relationship between reference values, free trajectory alignments and model fit better, we also determined the MSE parameters for trajectories with fixed stable values and constrained the alignments to those generated by free trajectories. Similar to the values in Table 5.4, Table 5.5 shows the $GMSE_{diphone}$ values, now with free trajectory alignments. As expected, the overall MSE measurements showed increased error for constrained alignments. Since the 3-piece model change descriptors depend so heavily on stable values, we found substantial error increases compared to the values of Table 5.5.

Model	Fixed stable value	MSE test	MSE stable values	MSE change descriptors
3-piece (utt)	Monophone	36.256 (8.871)	38.929 (9.911)	26.536 (7.305)
	Biphone	30.203 (5.964)	32.437 (6.723)	22.081 (5.063)
	Triphone	29.248 (5.583)	31.450 (6.215)	21.075 (4.915)
4-piece (utt)	Monophone	22.796 (5.051)	35.010 (9.054)	11.050 (2.228)
	Biphone	19.026 (3.180)	28.937 (5.542)	9.412 (1.513)
	Triphone	18.495 (3.085)	28.083 (4.956)	9.144 (1.569)

TABLE 5.5: *Overall $GMSE_{diphone}$ measurement on test set, when applying fixed stable values and free trajectory alignments.*

5.5.5 4-piece segments

In 3-piece models, a change descriptor consists of a single straight line, connected to the start and end points of the stable values (at the model alignments). To compare the effect of increased change descriptor complexity, the results of the previous section also include 4-piece segments. These models can model change by using two straight lines (see Section 5.4.2).

Comparing the overall consistency $G\hat{\sigma}_{trans}$ of the change descriptor position with that of the 3-piece models shows that the 4-piece models are the more consistent choice for free trajectory models (Table 5.4). Furthermore, rather than becoming less consistent when trajectories with fixed stable values are used, 4-piece models show comparable consistency for fixed stable value trajectories.

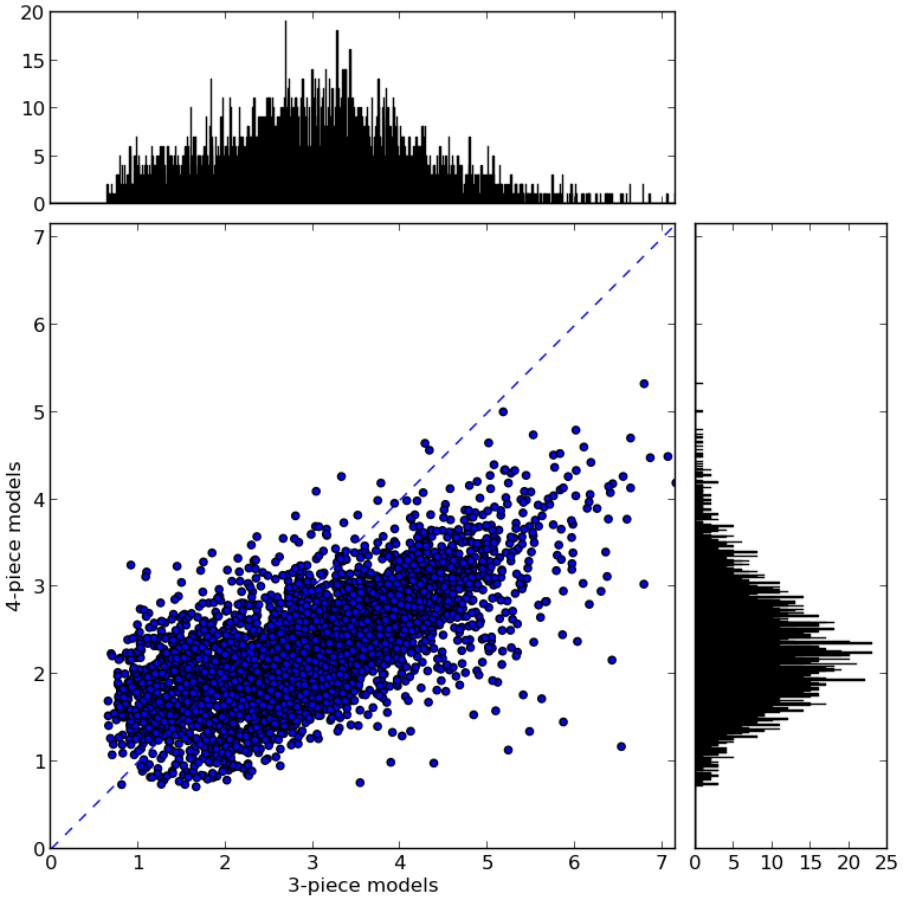

FIGURE 5.7: *Comparing consistency $\hat{\sigma}_{trans}$ of change descriptor position on a per-cepstrum basis for free alignments. It is clear that most cepstral transitions have larger standard deviations using 3-piece models, given the depicted histograms (top and right sides of the figure) of the data.*

More detailed comparisons can be obtained (on a per-channel basis) comparing the standard deviation $\hat{\sigma}_{trans}$ for the same cepstral transitions. Figure 5.7 shows how T_{mid} is measured relative to the ASR boundary for each cepstral transition example. Computing the standard deviation $\hat{\sigma}_{trans}$ allows transitions to be compared on a per-cepstrum basis. The scatter plot depicts these values for 3- and 4-piece models and the same transition examples. We found that most of the cepstral transitions had larger standard deviations when 3-piece models were used.

According to the frequencies and the placement of cepstral transition measurements in the histogram, only a relatively small number of cepstral transitions have a smaller standard deviation for 3-piece models. Generally 4-piece models also tend to have a lower standard deviation for most of these cepstral transitions.

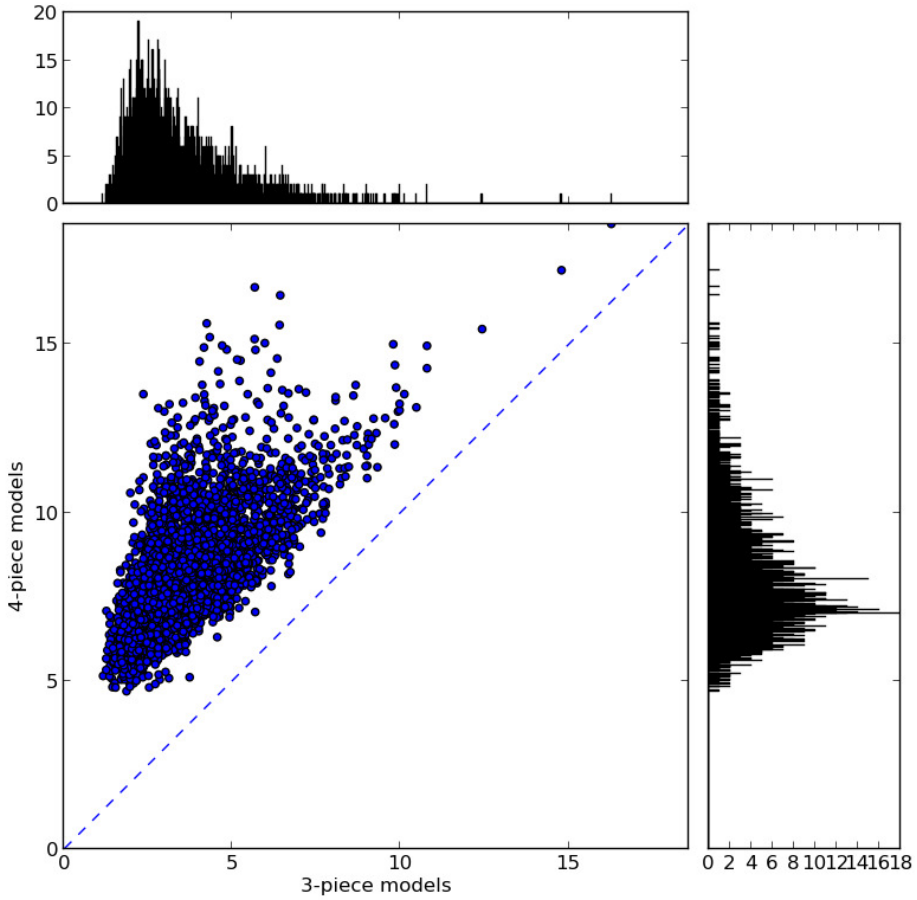

FIGURE 5.8: *Comparing mean duration $\hat{\mu}(p_{cs}, \omega)$ of the change descriptors on a per-cepstrum basis for free alignments. Histograms (top and right sides of the figure) clearly show the longer mean duration of change descriptors for 4-piece models.*

To gain better understanding of the differences between the 3-piece and 4-piece change descriptors, we also compared their absolute durations, $T_{dur} = T2 - T1$ (length in frames). Figure 5.8 shows the mean duration in frames compared for every cepstral transition class between 3-piece and 4-piece models. It is clear that for all cepstral transition classes, the mean durations of the change descriptors are longer for 4-piece models than for 3-piece models of the same class.

By confirming the overall variability $G\hat{\sigma}_{trans}$ in the mean duration for free trajectories (*Consistency of* T_{dur}), we found that connecting segments for 3-piece models seemed to provide the most consistent mean change descriptor durations in general (Table 5.6).

Model	**Consistency of T_{dur}**
3-piece (trans)	2.710 (0.858)
3-piece (utt)	2.583 (0.876)
4-piece (utt)	2.842 (0.930)

TABLE 5.6: *Overall consistency $G\hat{\sigma}_{trans}$ measurement of change descriptor durations (T_{dur}) on all data for free alignments showing the most consistent mean change descriptor durations for 3-piece models with connected segments.*

Although 4-piece models with longer change descriptors were less consistent, the values in the table indicate that this mean duration was still comparable to those of the 3-piece model.

Finally, the overall $GMSE_{diphone}$ values in Table 5.2 confirm that the additional freedom of the 4-piece model reduces the overall error significantly. The reduction is consistent for the error of all model parts, as well as these same measurements with utterance models, which incorporate predicted stable value parts, in comparison to the 3-piece models of similar configuration.

5.6 Conclusion

Piecewise linear transition models are able to represent phone transitions for ASR features, but these features do contain a variability that does not suggest a linear assumption for each feature transition. Low-pass filtering may be a means to improve this situation, by removing the non-linearity that the high-frequency components introduce (see Section 2.4.1). The next chapter investigates whether feature optimisation can strengthen feature trajectory extraction and what implications this holds for keeping ASR accuracy intact.

Chapter 6

Speech encoding: feature optimisation

6.1 Introduction

This chapter specifically describes the features that were used to encode speech data. Among the most widely used features for ASR applications are MFCCs. We chose to use this feature description as the basis (starting point) for modelling, tracking and ultimately constructing new trajectory-based ASR features. More specifically, the aim was to find a suitable procedure (consisting of various transformation operations) to enable the construction of specialised MFCC features suitable for trajectory modelling. At the same time, given a segment of speech, the feature construction process should allow the explicit modelling of frame-based trajectories.

MFCCs are generated from the outputs of an initial filter bank analysis, providing two widely differing levels of analysis. Section 6.3 compares the appropriateness of extracting features at either the spectral (filter bank) or cepstral (MFCC) level. At this stage, the comparisons we made were based solely on the model error (Section 6.3.4).

The approach to modelling speech trajectories in this study (as described in Chapter 5) required phoneme-level speech segmentation. (Phoneme-based time alignments are required prior to trajectory modelling.) Initially, this segmentation was obtained from known transcriptions of the test ("oracle" segmentation), but later the oracle segmentation was replaced by recognition-based alignments, which did not incorporate any information about the test set transcriptions. The data sets used as described in this chapter are explained in Section 6.3.2, and the metrics evaluated are described in Section 6.3.

In order to select appropriate feature parameter settings, we compared two different approaches to feature extraction: (1) variants of MFCC extraction, and (2) applying trajectory models during feature extraction. By comparing the phone recognition performance of (1) and (2) we could determine whether the information captured by the combination of features and trajectory models would suffice for ASR purposes.

Specifically, we considered parameters such as frame rate, spectral granularity, cepstral granularity and mean normalisation, and their effect on ASR accuracy. In addition, since recent work in noise-robust speech recognition has identified the reduction of feature variance as a crucial step during feature extraction, we also considered different smoothing approaches (see Section 6.5). Up to this point, analysis was performed using oracle alignments. Finally, as presented in Section 6.6, the oracle alignments were replaced by recognition-based alignments and the results evaluated.

6.2 Terminology

Table 6.1 briefly lists all of the terms and definitions used in this chapter to define all the different feature specifications.

Term	Interpretation
Trajectory-based ASR features	Specialised MFCC features constructed from trajectory models.
Feature channel	A one-dimensional set of sample values.
Intermediate features	A set of feature channels obtained during the feature construction process and before obtaining the final MFCC features.
Filter bank outputs	Sampled output of the triangular filters which is convolved with discrete Fourier transform (DFT) magnitude coefficients.
Initial cepstra	The cepstral coefficients generated directly as a function of the DCT.
Frame rate	The sample rate of a specific feature channel or set of feature channels.
Frame-based trajectories	A collection of feature channels, which are modelled to analyse trajectory information.
Spectral features	Feature channels of the filter bank outputs.
Cepstral features	Feature channels of MFCCs.
Trajectory models	Piecewise linear models that capture phone transition information.
Frame-based filtering	Low-pass (ARMA) filtering (smoothing) of intermediate features.

TABLE 6.1: *Terminology used throughout this work.*

6.3 Selecting level of analysis

The approach was to model the speech trajectories directly as a set of frame-based trajectories. We attempted to model frame-based trajectories by using piecewise linear

models. These linear trajectory models required slow-varying feature magnitudes that correspond with phone transitions to be most effective.

This section first describes how standard MFCCs were constructed, before considering where in this process the trajectories themselves could best be modelled. We analysed this question by modelling trajectories by means of features constructed in different ways, and evaluated the ability of the resulting models to approximate the actual data.

The data sets used for this analysis are described in Section 6.3.2. A phone-level segmentation of the speech data was required before we could create the trajectory models. This was not a straightforward process, as described in Section 6.3.3. We used two measures in order to evaluate the resulting models: (1) model error and (2) model-feature correlation. These are described in more detail in Section 6.3.4. Based on these measurements, a decision was made about the level of trajectory analysis that would be most suitable for use in the construction of the trajectory-based ASR features described in the subsequent sections.

6.3.1 MFCCs as starting point

The standard MFCC feature construction process can be summarised by the following main operations [92]:

1. Apply first-order pre-emphasis to the resulting signal and measure the magnitude.
2. Perform a fast Fourier transform (FFT) with a Hamming window on the speech signal.
3. Convolve the DFT magnitude coefficients with a number of triangular filters spaced to give approximately equal resolution on the mel-scale.
4. Apply a DCT to the n filter bank channels and obtain c cepstral coefficients.
5. Find the first- and second-order time derivatives of the cepstral coefficients and add these as additional coefficients.

Given the above description, there were two places of specific interest where frame-based trajectories could easily be extracted. These features carried the time-based trajectory information. The first place was right after the third step, when we extracted the raw filter bank outputs. (These spectral features were easily converted to a more linear format – in time – via the log operation.) Another possibility may be to consider the cepstral features that are formed at the end of the fourth operation and after the

application of the DCT. Previous work has shown that linear components (in time) are present in these frame-based trajectories [94]. Still, the DCT used for generating standard MFCCs could suppress time-based trajectories for individual feature channels. (The DCT combines a larger number of signal channels c in a complex way to generate the n cepstral channels.)

The linear trajectory model requires feature magnitudes to remain relatively stable between (before and after) phone transitions. Even when making the assumption that these linear components do dominate the information contained in the initial cepstra, taking the derivatives (during the last operation) is not expected to yield frame-based trajectories suitable enough to be modelled in this way. Also, if the key information can be captured adequately, there is no need for the additional modelling of derivatives. Given these considerations, we chose only to model the log filter bank outputs and the initial cepstra directly, using the linear trajectory models as discussed in Chapter 5.

6.3.2 Experimental data

The remainder of this chapter describes the different data sets used to conduct experiments. All the end results are presented in terms of the master train data and test data sets first described in Section 4.3. This set contains 4 072 training utterances and 902 test utterances. In addition, where we required a development data set (for parameter selection), this set of 873 utterances was selected from the 4 072 training utterances, using the same strategy as that for master train and test set partitioning. Less training data (a total of 3 199 utterances) for all phone recognition results is therefore reported for the development set.

6.3.3 Segmentation of speech data

As described above, a first segmentation of the speech data into phone units was obtained by using known transcriptions of the test data (oracle segmentation) during the initial analysis. This was later replaced by recognition-based alignments. A standard HMM-based ASR system was used in order to perform the segmentation. Phone transitions (as represented by the models in this study) were extracted as diphones, given this initial segmentation. The ASR system was trained on all 4 974 recordings. Since all recordings were used, it is possible that this alignment strategy could result in a bias toward the test data. For the experiments to follow, we expected the difference of results derived using a segmentation based only on the training data set to be minimal. In Section 7.9 we verified this expectation to be correct.

A context-dependent cross-word phone recogniser with tied-triphone models was built. We used a total of 39 MFCC features, including the first- and second-order derivatives of the first 13 cepstra. Each frame was computed with a window of 25 ms and the frame rate kept at 10 ms time intervals. The triphone models had three emitting states and seven Gaussian mixtures per state. Diagonal covariance matrices were used and finally semi-tied transforms were applied. Automatic alignments were obtained for triphones using a forced alignment of all the data, then the model alignment labels were converted back to the base label sequence (the actual phonemes observed in the training data).

Verifying flat phone recognition accuracy for a system with exactly the same configuration, but only trained on the 4 072 training utterances, yielded a value of 92.91% (as indicated in Table 6.3).

6.3.4 Measures of approximation efficiency

Fitting a trajectory model to a feature channel will approximate the value (magnitude) of the feature channel, which can in turn be sampled at any particular index in time. The "goodness of approximation" can then be measured simply by estimating the MSE between the original feature frames and the corresponding model values. Since the direct comparison of the error was required to motivate the choice of whether to use spectral features or cepstral features, the adequate normalisation of the feature channel's mean and variance was crucial.

Such a normalisation strategy can be implemented at different places in the feature construction or modelling process. Another option would be to normalise the differences in the shape of the feature distribution as part of the measurement strategy. Both options require a separate data set (to obtain objective estimates). We chose to normalise the feature channel mean separately, right at the start of the feature construction process. The mean was subtracted from the raw filter bank outputs on a per-utterance basis (see Section 6.4.1). Variance normalisation was achieved as part of the measurement strategy (we define a variance-weighted MSE measurement in Section 6.3.4.1) and at the cepstral level (where all cepstra have a normalised mean and variance).

The weighted MSE provides directly comparable estimates between feature channels (of different types) if the assumption is that the per-channel distribution of the features is normal. This assumption may be relaxed by considering the model-feature correlation instead. Since these values are independent of the feature distribution, no further normalisation is required either. We used the Pearson correlation coefficient as a secondary measure of similarity, which is described fully in Section 6.3.4.2.

The remainder of this section explains how we re-used a number of the terms defined in earlier chapters (Sections 5.5.1.2, 5.3.1 and 6.3.4.2), specifically:

- The sample standard deviation of parameter p across all S samples within the set ω:

$$\hat{\sigma}(p,\omega) = \sqrt{\frac{1}{S}\sum_{s=1}^{S}(p_s - \hat{\mu}(p,\omega))^2} \tag{6.1}$$

- The MSE between the actual (p) and modelled ($\hat{p}$) parameter values measured across all S samples within the set ω:

$$MSE(p,\hat{p},\omega) = \frac{1}{S}\sum_{s=1}^{S}(p_s - \hat{p}_s)^2 \tag{6.2}$$

- The sample Pearson correlation coefficient between the actual (p) and modelled ($\hat{p}$) parameter values measured across all S samples with the set ω:

$$r(p,\hat{p},\omega) = \frac{\sum_{s=1}^{S}(p_s - \hat{\mu}(p))(\hat{p}_s - \hat{\mu}(\hat{p}))}{\sqrt{\sum_{s=1}^{S}(p_s - \hat{\mu}(p))^2}\sqrt{\sum_{s=1}^{S}(\hat{p}_s - \hat{\mu}(\hat{p}))^2}} \tag{6.3}$$

6.3.4.1 Variance-weighted MSE

Using the definitions above, one can now calculate the variance-weighted mean square error $WMSE_{channel}$. In particular, let $\sigma^2(p,\omega_c)$ be the standard deviation of all frames (all utterances) for a particular feature channel c, then:

$$WMSE_{channel}(c) = \frac{MSE(p,\hat{p},\omega_c)}{\sigma^2(p,\omega_c)} \text{ where } \omega_c = \bigcup_{f=1}^{F_c}(p_{fc},\hat{p}_{fc}) \tag{6.4}$$

with F_c the number of frames f observed in the channel. Taking the mean of the individual $WMSE_{channel}$ values for all feature channels C then yields the final variance-weighted MSE value ($GWMSE_{channel}$):

$$GWMSE_{channel} = \frac{1}{C}\sum_{c=1}^{C}WMSE_{channel}(c) \tag{6.5}$$

for all feature channels c.

6.3.4.2 Correlation measure

As a secondary measure, the Pearson correlation coefficient $r_{channel}$ can be calculated for each channel c:

$$r_{channel}(c) \quad = \quad r(p, \hat{p}, \omega_c) \text{ where } \omega_c = \bigcup_{f=1}^{F_c} (p_{fc}, \hat{p}_{fc}) \tag{6.6}$$

Similarly, a global estimate $Gr_{channel}$ yields a single correlation value (the mean) across all feature channels:

$$Gr_{channel} \quad = \quad \frac{1}{C} \sum_{c=1}^{C} r_{channel}(c) \tag{6.7}$$

6.3.5 Comparing model approximation

As discussed (Section 6.3.4), the variance-weighted MSE and the Pearson correlation for a set of frame-based trajectories can be used as direct indicators to model efficiency. Since we were particularly interested in testing how spectral and cepstral features compared in this regard, we chose to model both of these intermediate feature types for the master test data set. This comparison was conducted by using the 3-piece linear models defined previously (see Sections 5.3 and 5.4.1) and modelling the trajectories of all feature channels in this way. Also, we initially kept typical values for the parameters such as the frame rate, number of filters in the filter bank and the number of cepstra generated. These values were: a 10 ms frame rate, 26 filter bank channels and 13 MFCCs respectively.

Specifically, we realise that MSE and correlation measures inherently favour smoothed trajectories - they would need to be balanced by a measure of discrimination for a valid comparison. This follows later in Section 6.5.2 when evaluating the effect of ARMA filtering on these measures and including phone recognition accuracies. That more extensive evaluation supports the MSE and correlation measures as a valid comparison. We do not perform phone recognition for each assessment, since the phone recognition evaluation task itself is much more costly to compute than MSE and correlation measures. The values for the $GWMSE_{channel}$ (MSE) and $Gr_{channel}$ (Cor) for the different feature options are specifically compared in Table 6.2. In addition, the table specifies the type of features, number of channels and the total number of frames for each result.

No	Feature type	Channels	Frame rate	MSE	Cor	No of values
1	Cepstral	13	10ms	0.1549	0.9187	2 949 739
2	Cepstral	13	5ms	0.1905	0.8986	5 653 804
3	Spectral	20	5ms	0.0534	0.9729	8 698 320
4	Spectral	26	10ms	0.0593	0.9699	5 899 478
5	Spectral	26	5ms	0.0578	0.9706	11 307 816
6	Spectral (dev)	26	5ms	0.0576	0.9708	11 026 990
7	Spectral	26	2.5ms	0.0570	0.9710	22 615 632
8	Spectral	39	5ms	0.0659	0.9665	16 961 724

TABLE 6.2: *Comparing the weighted MSE and correlation measurements for trajectory models of different intermediate features in the master test data set showing higher model error for cepstral features than when using spectral features.*

The MSE values listed in Table 6.2 for this particular comparison clearly show that the model-feature mismatch is much higher for cepstral features than for spectral features. Similarly, model-feature correlation is lower for cepstral features and, considering all the results, one can see exactly the same trends for the correlation measurements and MSE measurements. Figure 6.1 shows just how similar these measurements are. It is clear that the shape of the feature channel distributions did not affect the MSE measurement in these tests.

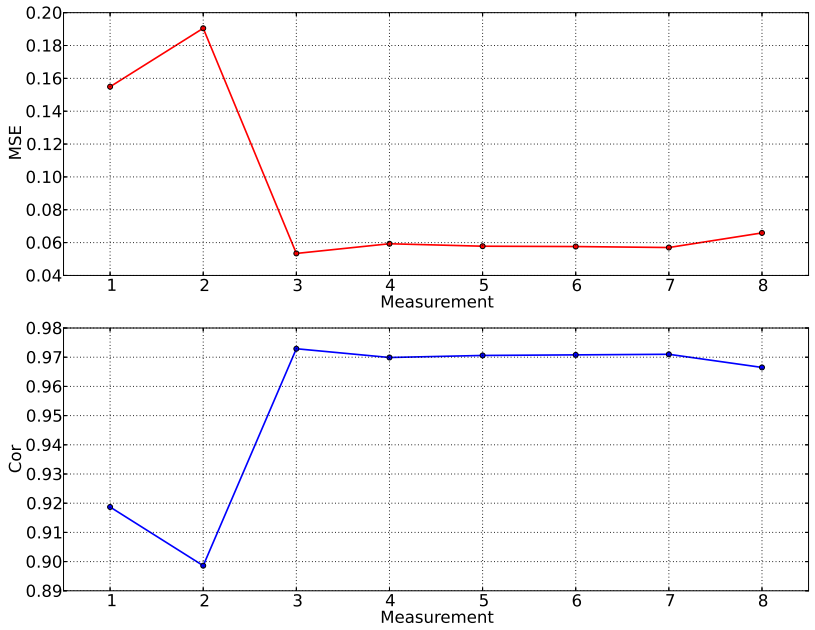

FIGURE 6.1: *Comparing the similarity of correlation and MSE measures regarding estimated model error (Table 6.2) across both spectral and cepstral features indicate unaffected MSE measurement for feature channel distribution shape.*

The finding that model-feature mismatch is much higher for cepstral features than for spectral features might be due to several factors. In the DCT operation there were

26 filter bank channels that were combined to form 13 cepstra. This combination of feature magnitudes alone gave rise to more non-linear behaviour in the cepstral features. Another consideration is the issue of granularity. About $3 * 10^6$ feature values were used for cepstral model estimation for the test data set and double this number for spectral model estimation, which in turn allowed twice the number of linear models to be used. Here, more feature values did not directly translate to more data, since the spectral feature channels were correlated. The cepstral and spectral features were just different representations of the same data. However, it turned out that spectral features fit the piecewise linear modelling structure we applied better. For these reasons, we chose to use only the spectral models from this point onward. Given this choice, we also added an MSE measurement for the development data set (denoted as dev) where 26 filters were used at a frame rate of 5 ms.

6.4 Feature-parameter analysis

This section details the extension of the standard MFCC feature description, introducing an elaborate feature construction procedure (front-end). What is important is that the additional operations included the option of the explicit modelling of frame-based trajectories and the construction of specialised MFCCs based solely on these models. Section 6.4.1 describes this procedure in detail. To evaluate the newly formed features, we built phone recognition systems and tested the flat phone accuracies. Several experiments were conducted, establishing the control measurements (Section 6.4.2), the impact of frame rate (Section 6.4.4) and the number of filter bank channels to be used (Section 6.4.3). In addition, the testing of the granularity (frame rate) for model estimation was explained in Section 6.4.3.

6.4.1 Feature construction

As mentioned in 6.1, the end goal of the ASR front-ends we defined was to generate specialised MFCC features. Since additional steps, such as filtering or the approximation of spectral features, might be activated, the resulting ASR features may not be typical MFCCs. The following sections discuss how we conducted tests to establish whether including the required approximations as part of the front-end could yield acceptable phone recognition accuracies. Figure 6.2 is a block diagram of the new set of components for the ASR front-end used in the remainder of this chapter. The detailed procedure to generate these specialised MFCCs is summarised in the following steps:

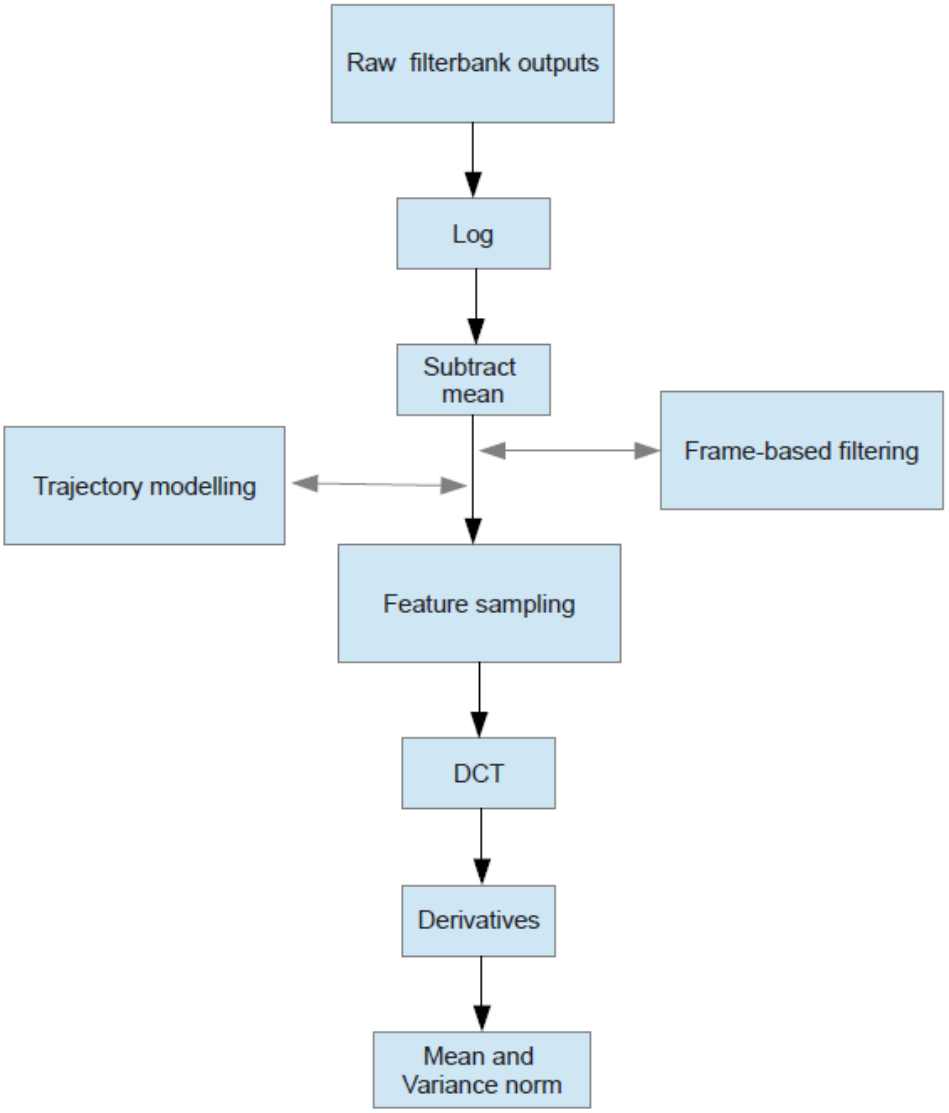

FIGURE 6.2: *Feature construction procedure optionally including trajectory modelling and frame-based filtering to create specialised MFCCs.*

1. Obtain raw filter bank outputs: A Hamming window with a pre-emphasis coefficient of 0.97 and a window size of 25 ms is used to perform an FFT. Only frequencies in the range of 150–8 000 Hz are allowed. Depending on the experiment, by using this configuration the speech signal is sampled at a frame rate of 2.5–15 ms. In the same manner (as discussed in Section 6.3.5) the number of filters in the filter bank may be set at between 20–39 filters.

2. Convert to log filter bank outputs: the log operation is applied to the intermediate feature values.

3. Perform mean subtraction: for each feature channel, the mean is subtracted on a per-utterance basis.

4. Frame-based filtering: optional frame-based smoothing (low-pass filtering) is applied to obtain a new set of features. Moving average (MA) or ARMA filters are used (Section 6.5).

5. Trajectory modelling: a trajectory model may be estimated for each feature channel. The trajectory models can then serve as a new set of intermediate features.

6. Feature sampling: a particular frame rate is required for the cepstral features. The previous features are re-sampled to build this intermediate feature set. If these previous features are given in terms of trajectory models, sampling will return the value of the trajectory model at each required time step.

7. Apply DCT: the log operation (of step 2) is reversed if this is required. We applied the DCT and obtained 13 MFCCs, which included MFCC 0. Finally cepstral liftering as implemented in HTK [92] is used with a cepstral liftering coefficient of 22.

8. Add derivative features: obtain the first- and second-order derivatives of the first 13 specialised MFCCs using the function defined for HTK [92] and with a default window size of two frames.

9. Normalisation of features: apply the cepstral mean and variance normalisation (all utterances) to each of the 39 cepstra at this point.

6.4.2 Control measurements

The first priority was to obtain benchmark phone recognition accuracies for comparing the recognition results for the specialised front-ends tested in the experiments. Specifically, these phone recognition systems had the following characteristics:

- All MFCC features used for training acoustic models were created by using the defined feature construction procedure in 6.4.1 (note that no ARMA filtering was used for control measures).

- The acoustic models were built with exactly the same parameters as the standard HMM system that provided the initial segmentation of the speech data (described in Section 6.3.3).

- ASR features for the test data were constructed in exactly the same manner as for the training data. Phone alignments were required when estimating the trajectory models (for the spectral features). We used the alignments obtained during the

initial speech segmentation (Section 6.3.3) in all the experiments discussed in Section 6.4. As explained, we present proof in Section 7.9 that including the test data in the initial segmentation process has a minimal effect on the results generated.

System	ACC	IP	ACC (semi-tied)	IP	MSE	Cor
Standard (segmentation)	92.91	-40	93.82	-35	-	-
Control	92.67	-40	93.90	-35	-	-
Control (no mean)	90.41	-40	91.33	-35	-	-
Linear	91.03	-41	92.79	-34	0.0578	0.9706
Linear (normal test)	79.75	-62	79.60	-58	0.0578	0.9706
Dev: Control	91.70	-40	92.99	-35	-	-
Dev: Linear	89.46	-41	91.46	-34	0.0576	0.9708
Dev: Linear (normal test)	76.87	-62	74.42	-58	0.0576	0.9708

TABLE 6.3: *Baseline phone recognition results for systems trained with standard MFCCs and specialised MFCCs for which a few key options are systematically activated. The bottom part of the table includes results for systems used to tune the insertion penalty value.*

Table 6.3 compares the flat phone recognition accuracies for a number of systems. In addition, the table states the phone insertion penalty (IP) value at which each individual accuracy was obtained. A search of the insertion penalty values (performed at unit intervals) was used to find the best accuracy for every measurement in the development test data (see Section 6.5.2). Two accuracies are given for each evaluation. The second and better accuracy is achieved when a semi-tied transform is also applied. Finally the $GWMSE_{channel}$ and $Gr_{channel}$ values are given to quantify the model-feature difference for the linear systems. A detailed system description is as follows:

- **Standard**: a phone recogniser with exactly the same parameters and front-end as the system that provided the initial segmentation of the speech data (see Section 6.3.3).
- **Control**: the same as the standard system, but using the front-end as defined in Section 6.4.1.
- **Control (no mean)**: equivalent to the control system, but no feature mean or variance normalisation is activated.
- **Linear**: the control system configuration and activating linear trajectory models as an intermediate feature set in the front-end.
- **Linear (normal test)**: the same acoustic models as for the linear system are used, but the test features of the control system are tested. Trajectory model features are not estimated for the test set.

As expected, the standard and control systems were verified to yield the highest phone accuracies (93.82% and 93.90% respectively). For this (single speaker) dataset, the slight difference (more noticeable without the semi-tied transform) was due to the variance normalisation of the cepstral features for the control system. The per-utterance mean subtraction contributed substantially to this. Recognition accuracy dropped significantly (to 91.33%) for the control system where all mean and variance normalisation was deactivated (no mean). By comparison, constructing specialised MFCC features from trajectory models in the front-end performed exceptionally well. A phone accuracy as high as 92.79% was obtained. These trends were verified by using a development set, from which the phone insertion penalties were obtained. Recognition accuracies for these experiments were slightly lower; the linear system had an accuracy of 91.46%. There are two reasons for the lower accuracies obtained. We pointed out in Section 6.3.2 that the development data set is kept completely separate from the test data set and consequently we trained the acoustic models for the development set evaluation on less training data (3 199 utterances). Finally, it was clear that merely using the control system's test features without modification introduced significant mismatch for the acoustic models trained with specialised MFCCs where trajectory models were intermediate features.

6.4.3 Granularity of spectral analysis

The trajectory models that were used essentially function at a phone transitional level. As discussed in Chapter 5, we searched to obtain an additional pair of model alignments for every diphone transition and feature channel. In doing so, each model segment could be seen as a detector to identify a single area of change (modelled by the change descriptor). Since diphone transitions are relatively short units in time, only a limited number of frames are available to achieve adequate model estimation. The number of frames for the intermediate features from which the trajectory models were estimated corresponded directly to the rate at which the filter bank was sampled.

We verified the effect that the filter bank frame rate had on phone recognition accuracy for systems based on linear trajectory models. Frame rates of 2.5–10 ms were tested. To this end, Table 6.4 lists the flat phone recognition results for three systems with different filter bank frame rates on the master test data. Insertion penalties were obtained, using the optimum result for the development test set of the 5 ms system (as in Table 6.3). The $GWMSE_{channel}$ and $Gr_{channel}$ measurements were also included.

We found that phone recognition accuracy did improve when using twice the number of frames (5 ms frame rate) for model estimation. Only slight differences could be noted,

System	Frame rate (filter bank)	ACC	IP	ACC (semi-tied)	IP	MSE	Cor
Linear	2.5 ms	91.15	-41	91.17	-34	0.0570	0.9710
Linear	5 ms	91.03	-41	92.79	-34	0.0578	0.9706
Linear	10 ms	89.78	-41	91.74	-34	0.0593	0.9699

TABLE 6.4: *Effect of frame rate on phone recognition accuracy when the spectral features are linear trajectory models.*

given the MSE and correlation measurements. Although these values were better for the 2.5 ms frame rate, no further improvement was detected when activating semi-tied transforms. Similarly, the number of filters used determined the bandwidth that each filter (feature channel) encoded. The spectral resolution might be increased by using a larger number of filters. The literature on filter bank design suggests that 20 and 26 filters on the mel-scale would be optimal for MFCC front-ends. Increasing the granularity might lead to less variation per channel, since the same segment of speech would be modelled by more trajectory models. Table 6.5 includes the MSE and correlation measurements for three trajectory-based recognition systems that were explored. We trained these systems on features that had 20, 26 and 39 channels respectively. A control system (without using trajectory models as features) is also given for each trajectory-based system (Linear), and phone insertion penalties are taken from the 26 channel systems on the development test set (as in Table 6.3).

System	Channels	ACC	IP	ACC (semi-tied)	IP	MSE	Cor
Control	20	92.54	-40	93.61	-35	-	-
Linear	20	90.15	-41	92.25	-34	0.0534	0.9729
Control	26	92.67	-40	93.90	-35	-	-
Linear	26	91.03	-41	92.79	-34	0.0578	0.9706
Control	39	92.61	-40	93.82	-35	-	-
Linear	39	91.16	-41	92.73	-34	0.0659	0.9665

TABLE 6.5: *Effect of different numbers of filter bank channels on the phone recognition results for control and trajectory-based (linear) systems.*

Comparing the model approximation (MSE and Cor values) indicates that only slight differences are detected near the standard number of channels. These differences seem to indicate that better model approximation is achieved when using fewer channels. Evaluating the phone recognition accuracies does show, however, that the best accuracy is obtained with 26 filters, both for the control and for the linear systems. The result for systems with 39 filters was not found to provide any further improvement, nor when testing the system employing trajectory models as the intermediate features.

6.4.4 Impact of frame rate

Section 6.4.1 states that re-sampling the spectral representation was performed at the frame rate used when constructing the cepstral features. The typical frame rate for constructing the cepstral features of MFCC-based recognition systems is around a value of 10 ms (see [92]). This section explains how we tested whether the modified ASR front-ends we used require a similar setting for this parameter. Flat phone recognition accuracies are listed in Table 6.6 for frame rates of 5 ms, 10 ms and 15 ms respectively and we chose insertion penalties based on the development set test of the 10 ms systems (see Table 6.3). Again, both systems were compared, using trajectory models (Linear) and without using trajectory models (Control). The values clearly show that the best recognition accuracy is achieved (with and without semi-tied transforms) at a frame rate of 10 ms for all systems.

System	Frame rate	ACC	IP	ACC (semi-tied)	IP
Control	5 ms	89.81	-40	90.19	-35
Linear	5 ms	85.75	-41	86.76	-34
Control	10 ms	92.67	-40	93.90	-35
Linear	10 ms	91.03	-41	92.79	-34
Control	15 ms	91.03	-40	92.59	-35
Linear	15 ms	88.65	-41	90.33	-34

TABLE 6.6: *Effect of different frame rates on phone recognition results for control and trajectory-based (Linear) systems*

6.5 Reducing the feature variance (low-pass filtering)

Phone recognition accuracy may be improved by the additional adjustment of the feature variance. Moreover, it has been found that low-pass filtering has to be an integral part of approaches to reduce acoustic mismatch between clean and noisy speech data (see Section 2.3.1.2). Since the purpose of the trajectory models was to capture the "slow" and most prominent changes during phone transitions, higher frequency components were already being discarded. Reducing the unwanted (high-frequency) components may therefore improve the accuracy with which the prominent change components can be detected. To test this hypothesis, we conducted several recognition-based experiments and evaluated the effect of MA or ARMA filtering on the spectral features of the ASR front-end (Section 6.5.2). We used a constrained form of these filters, only searching through a range of filter orders to find a balance between feature smoothing and accurate detection of phone transitions. This strategy proved sufficient during the analysis but allowing the filter parameters to vary should permit more freedom to improve feature smoothing. Throughout the experiments, we tracked the model-feature mismatch

for the trajectory models, using the previously defined $GWMSE_{channel}$ and $Gr_{channel}$ measurements. We report below on the observed changes and discuss the implications these changes have for phone recognition accuracy (Section 6.5.3).

6.5.1 Frame-based smoothing

To implement an MA filter (of a particular order) for a feature channel, let X be the feature vector and $M(X_i)$ be the MA post-processed feature at frame index i. Then the MA filter is defined by:

$$M(X_i) = \frac{X_{(i-m)} + ... + X_{(i-1)} + X_i + ... + X_{(i+m)}}{2m+1} \tag{6.8}$$

where m is the order of the MA filter. The first and the last frames are duplicated as needed at the beginning and the end of the frame sequence to facilitate the filter order.

Equation 6.8 is easily extended to accommodate additional capability for auto-regressive feature estimation. Let X be the feature vector and $R(i)$ be the ARMA post-processed feature at frame index i respectively. Then the ARMA filter is defined by:

$$A(X_i) = \frac{R_{(i-m)} + ... + R_{(i-1)} + X_i + ... + X_{(i+m)}}{2m+1} \tag{6.9}$$

where m is the order of the ARMA filter. Consequently, the special case of $m = 0$ degenerates to no filtering.

6.5.2 Phone recognition accuracy

We found it useful to investigate first the effect that the smoothing of feature channels had on phone accuracy, before considering its effect on trajectory model estimation. As before, these systems are referred to in Tables 6.7 and 6.8 as the "control". The level of smoothness was adjusted by means of controlling the filter order (changing the number of frames used for generating each individual frame of the new feature set). The tables indicate the filter type (filter) for each system with the filter order in brackets. The flat phone recognition accuracies (ACC) were provided for the insertion penalties we found for the development set experiments and ARMA filters. (The insertion penalty of the ARMA filter of the same order was used for the MA filters.) Finally we added the MSE and correlation (Cor) measurements as had been done in all previous analyses to track the model-feature mismatch.

System	Filter	ACC	IP	ACC (semi-tied)	IP	MSE	Cor
Control	ARMA (1)	91.76	-42	92.90	-33	-	-
Linear	ARMA (1)	90.10	-42	90.79	-48	0.0422	0.9787
Control	ARMA (2)	91.83	-48	92.94	-36	-	-
Linear	ARMA (2)	90.96	-44	92.08	-41	0.0325	0.9836
Control	ARMA (3)	92.05	-45	93.29	-42	-	-
Linear	ARMA (3)	90.89	-42	90.81	-46	0.0261	0.9868
Control	ARMA (4)	91.96	-43	93.18	-43	-	-
Linear	ARMA (4)	90.95	-45	92.29	-40	0.0216	0.9891
Control	ARMA (5)	91.94	-50	93.00	-35	-	-
Linear	ARMA (5)	91.31	-47	92.60	-38	0.0183	0.9908
Control	ARMA (6)	91.93	-44	93.27	-35	-	-
Linear	ARMA (6)	91.51	-38	92.88	-35	0.0158	0.9920
Control	ARMA (7)	92.18	-41	93.19	-38	-	-
Linear	ARMA (7)	91.54	-41	92.92	-39	0.0139	0.9930
Control	ARMA (8)	92.08	-38	93.19	-42	-	-
Linear	ARMA (8)	91.57	-39	92.85	-32	0.0125	0.9937
Control	ARMA (10)	91.77	-42	93.07	-35	-	-
Linear	ARMA (10)	90.65	-34	92.20	-30	0.0102	0.9949

TABLE 6.7: *Development set: effect of ARMA filtering on the phone recognition results of the control and trajectory-based (Linear) systems for filters with different orders (Filter).*

We began by firstly considering the type of filter used for feature smoothing. As the MSE and correlation values in Table 6.8 indicate, ARMA filtering is more effective in generating features that are more linear (in time) than MA filtering of the same order. These "smoothed" features can therefore be better approximated by the linear trajectory models. Looking at the phone recognition accuracies, only slight differences in accuracy were observed for the "control" systems, with more fluctuations for lower-order filters It seems that phone recognition accuracies remain constant (near the optimal) for a broad range of higher filter orders (4 to 8) and using semi-tied transforms. Similarly, when semi-tied transforms are not used, a higher level of smoothness also seems to work, with good phone recognition results between the filter orders 4 and 8. From this point on, we chose to use only ARMA filtering.

As expected, the results for trajectory-based (Linear) systems do seem to be improved for greater levels of feature smoothness. We observed that the filter order has to be greater than a value of 3 to work adequately when using ARMA filtering. Similarly, the linear systems in the development set (Table 6.7) were consistent with the observed instability at lower orders of ARMA filtering. This effect may be due to the presence of model alignment error at low levels of smoothness.

In a trajectory-based system, the best performance (for the master test set) was obtained at an order of 6 (achieving an accuracy of 93.78%), after which the recognition accuracy gradually decreased again. (Development set tests reached a maximum accuracy of 92.92% at ARMA order 7.) Without semi-tied transforms, an optimal value was obtained

System	Filter	ACC	IP	ACC (semi-tied)	IP	MSE	Cor
Control	MA (1)	92.80	-42	94.05	-33	-	-
Linear	MA (1)	91.17	-42	92.48	-48	0.0503	0.9745
Control	ARMA (1)	92.71	-42	93.71	-33	-	-
Linear	ARMA (1)	91.58	-42	88.40	-48	0.0423	0.9786
Control	MA (2)	92.75	-48	93.86	-36	-	-
Linear	MA (2)	91.53	-44	92.65	-41	0.0384	0.9806
Control	ARMA (2)	92.71	-48	93.97	-36	-	-
Linear	ARMA (2)	92.07	-44	91.92	-41	0.0327	0.9835
Control	MA (3)	92.96	-45	93.95	-42	-	-
Linear	MA (3)	91.97	-42	93.65	-46	0.0299	0.9849
Control	ARMA (3)	92.05	-45	93.81	-42	-	-
Linear	ARMA (3)	92.19	-42	93.23	-46	0.0264	0.9867
Control	MA (4)	93.05	-43	93.99	-43	-	-
Linear	MA (4)	92.20	-45	93.59	-45	0.0239	0.9880
Control	ARMA (4)	93.08	-43	93.97	-43	-	-
Linear	ARMA (4)	92.32	-45	93.51	-45	0.0218	0.9890
Control	MA (5)	93.07	-50	93.82	-35	-	-
Linear	MA (5)	92.53	-47	93.49	-38	0.0196	0.9901
Control	ARMA (5)	93.00	-50	93.94	-35	-	-
Linear	ARMA (5)	92.35	-47	93.44	-38	0.0185	0.9907
Control	ARMA (6)	93.06	-44	93.97	-35	-	-
Linear	ARMA (6)	92.38	-38	93.78	-35	0.0160	0.9919
Control	ARMA (7)	93.01	-41	93.95	-38	-	-
Linear	ARMA (7)	92.55	-41	93.71	-39	0.0141	0.9929
Control	ARMA (8)	93.21	-38	93.92	-42	-	-
Linear	ARMA (8)	92.42	-39	93.62	-32	0.0126	0.9937
Control	ARMA (10)	92.81	-42	93.77	-35	-	-
Linear	ARMA (10)	92.25	-34	93.31	-30	0.0103	0.9948

TABLE 6.8: *Effect of MA and ARMA filtering on the phone recognition results of control and trajectory-based (linear) systems for filters with different orders (Filter). Correlation (Cor) and MSE measures clearly indicate ARMA filters to be more effective to generate smooth features which are better approximated by linear trajectory models. Phone recognition accuracies remain near optimal for higher filter orders (4 to 8) and using semi-tied transforms.*

for similar orders (between 6 and 8), confirming the improved linear model estimation at these higher levels of smoothness. A comparison of MSE model-feature values supports this claim. These values clearly showed the extent to which the model-feature mismatch was reduced when first applying ARMA frame-based filtering, before model estimation. Similarly, the same values shown in Table 6.7 demonstrate the stability of the MSE measurement.

Figure 6.3 provides several graphs of the phone recognition accuracy on the master test data, for the range -5 to -65 of insertion penalty values. It is clear that higher order ARMA filtering is a viable approach to ensure good accuracy with trajectory-based systems. The results of these systems also displayed a closely similar stability (across insertion penalties) compared to the control system. Lastly, Figure 6.3 confirms that a search of unit intervals on the insertion penalty values is sufficient when determining optimal accuracy in these experiments.

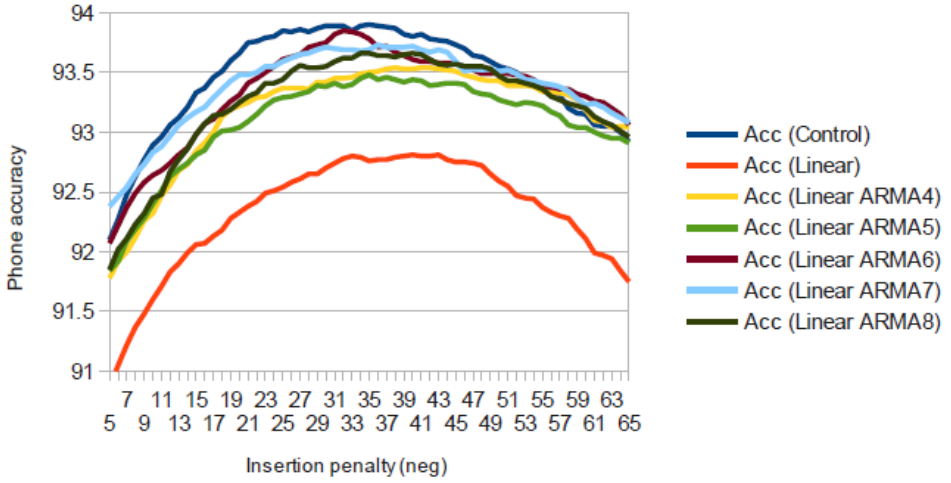

FIGURE 6.3: *Phone recognition stability of linear trajectory-based systems show that higher order ARMA filtering is a viable strategy to ensure good accuracy.*

6.5.3 Changes to model estimation

It is interesting that the model-feature mismatch (MSE) can be reduced by more than half an order in size and have the equivalent recognition systems still performing optimally (even better than the system without any variance adjustment). The reason for this is that the trajectory models created also have to be significantly different to accommodate the switch to smoothed features. To gain an additional perspective on this change, we estimated the $GWMSE_{channel}$ and $Gr_{channel}$ measurements (as defined in Section 6.3.4), but always used raw test features (without any applied filtering), to measure the MSE of these "control" features with trajectory models estimated at various levels of smoothness (ARMA filtering). Finally, the feature channel variance ($\sigma^2(p, \omega_c)$ in Section 6.3.4.1) that was used to normalise the feature channel variance was set to the variance of the control features as well.

The newly estimated $GWMSE_{channel}$ and $Gr_{channel}$ values for the master and development test data sets are listed in Table 6.9. At this point, it is interesting to note that the literature on noise robustness finds that an ARMA filter of about order 2 is optimal when reducing feature mismatch (see Section 2.3.1.2). Indeed, we found that the trajectory models estimated for features with a filter order of 2 had the lowest mismatch (close to the system where no filtering was applied) with the control features. As expected, we also measure increased differences between the original (Control) features and the trajectory models estimated for smoothed features for orders 3 and higher.

Filter	MSE	MSE (dev)	Cor	Cor (dev)
-	0.0578	0.0576	0.9706	0.9708
MA (1)	0.0597	-	0.9699	-
ARMA (1)	0.0647	0.0646	0.9671	0.9671
MA (2)	0.0595	-	0.9707	-
ARMA (2)	0.0617	0.0616	0.9686	0.9687
MA (3)	0.0619	-	0.9703	-
ARMA (3)	0.0618	0.0616	0.9688	0.9689
MA (4)	0.0665	-	0.9689	-
ARMA (4)	0.0642	0.0640	0.9679	0.9680
MA (5)	0.0733	-	0.9666	-
ARMA (5)	0.0686	0.0685	0.9659	0.9660
ARMA (6)	0.0748	0.0747	0.9631	0.9633
ARMA (7)	0.0825	0.0823	0.9596	0.9597
ARMA (8)	0.0914	0.0912	0.9554	0.9555
ARMA (10)	0.1124	0.1122	0.9450	0.9452

TABLE 6.9: *Effect of MA and ARMA filtering on model estimation detected by estimating MSE and correlation (Cor) measures for "control" features instead.*

6.6 Recognition with trajectory-based features

In all the previous sections, we constructed specialised MFCCs and used trajectory models as intermediate features during the feature construction process. The estimation of trajectory models requires accurate phone alignments in order to segment the speech and obtain the location of individual diphone transitions. Up to this point, we used the phone alignments obtained from the segmentation system as described in Section 6.3.3 for this purpose. In practice a test utterance does not have these alignments so they must be inferred automatically before any trajectory modelling (for test data) can be achieved. Next we investigated how best to achieve this outcome, given all the feature construction techniques we used during this analysis.

System	Test data	ACC	IP	ACC (semi-tied)	IP
Linear	Control	79.75	-62	79.60	-58
Linear (ARMA 6)	Control (ARMA 6)	88.44	-52	89.61	-51
Linear (ARMA 8)	Control (ARMA 8)	88.82	-49	89.99	-40
Dev: Linear	Control	76.87	-62	74.42	-58
Dev: Linear (ARMA 6)	Control (ARMA 6)	86.90	-52	88.77	-51
Dev: Linear (ARMA 8)	Control (ARMA 8)	87.76	-49	89.26	-40

TABLE 6.10: *Phone recognition results showing reduced mismatch for trajectory-based systems and filtered (smoothed) test features setting insertion penalties on the development set (Dev).*

Tables 6.10 and 6.11 show the recognition results that evaluate the system performance for trajectory-based (Linear) acoustic models and when merely using standard test features (Control) with the same level of smoothness (the order of ARMA filtering applied is shown in brackets). Phone recognition accuracies were optimised by using the development test set for systems at three different levels of smoothness. We activated the

ARMA filtering of orders 0, 6 and 8 respectively. As before, the values for these same tests at the determined insertion penalties are given for the master test data set (see the top part of the table).

It is clear that merely using the normal test features (without adjustment) leads to a significant mismatch. Table 6.10 indicates that a result of only 79.60% was achieved compared to 92.79% accuracy for the test data based on trajectory models (see Table 6.3). Since the previous sections state that promising results were obtained for smoothed features, we first tested to see whether this situation could be remedied by using equivalent levels of smoothing for the test data at higher orders of ARMA filtering.

System	Test data	ACC	IP	ACC (semi-tied)	IP
Linear	Control	79.86	-64	79.86	-64
Linear (ARMA 4)	Control (ARMA 4)	83.01	-79	82.54	-85
Linear (ARMA 5)	Control (ARMA 5)	82.83	-80	85.06	-70
Linear (ARMA 6)	Control (ARMA 6)	88.69	-64	89.69	-59
Linear (ARMA 7)	Control (ARMA 7)	88.93	-63	89.96	-54
Linear (ARMA 8)	Control (ARMA 8)	88.82	-49	90.31	-51
Linear (ARMA 10)	Control (ARMA 10)	88.97	-46	90.62	-50

TABLE 6.11: *Master test set with optimised insertion penalties (IP): phone recognition results showing reduced mismatch for trajectory-based systems and filtered (smoothed) test features across a broad range of filter orders.*

Table 6.11 gives the optimised phone recognition values for the master test data set (the results were evaluated for a broader range of ARMA filtering orders in this way). The results showed that considerable improvement could be obtained for high levels of smoothness. When using an ARMA filter of order 10, the optimised accuracy of the master test set was restored to a value as high as 90.62%. Table 6.10 shows that a phone accuracy of 89.99% is obtained for an ARMA filter of order 8, using the optimised insertion penalty of the development test set.

6.6.1 Conversion of trajectory-based test features

System	Test data	ACC	IP	ACC (semi-tied)	IP
Linear	Linear	88.97	-38	91.54	-38
Linear (ARMA 6)	Linear (ARMA 6)	91.07	-39	92.49	-37
Linear (ARMA 8)	Linear (ARMA 8)	91.02	-37	92.32	-37
Dev: Linear	Linear	87.44	-38	89.55	-38
Dev: Linear (ARMA 6)	Linear (ARMA 6)	89.85	-39	91.18	-37
Dev: Linear (ARMA 8)	Linear (ARMA 8)	89.87	-37	91.03	-37

TABLE 6.12: *Effect of converting test data to the same trajectory-based features using the phone alignments of the original recognition system, setting insertion penalties on the development set (Dev). Converting to trajectory-based front-end features provides accuracy close to the control.*

Although high levels of smoothness improved the results considerably, the best result was not yet a match for the trajectory-based systems that were tested as discussed in Section 6.5. Further adjustment of the test data was required, through the estimation of trajectory models for the intermediate test features. The fact that recognition accuracies (for systems built using front-ends with trajectory-based features) were so dependent on the level of smoothness alone was advantageous, however, in the sense that more accurate phone recognition alignment could possibly become tractable. We first established a gold standard to investigate the effect of alignment on trajectory model estimation for the test features. We set the phone alignments to be generated by the standard (see the "control" phone recognition system in Table 6.3) flat phone recogniser.

System	Test data	ACC	IP	ACC (semi-tied)	IP
Linear	Linear	89.11	-44	91.54	-38
Linear (ARMA 4)	Linear (ARMA 4)	91.14	-46	92.31	-40
Linear (ARMA 5)	Linear (ARMA 5)	91.38	-42	92.43	-33
Linear (ARMA 6)	Linear (ARMA 6)	91.32	-48	92.53	-39
Linear (ARMA 7)	Linear (ARMA 7)	91.40	-48	92.54	-42
Linear (ARMA 8)	Linear (ARMA 8)	91.10	-43	92.43	-44
Linear (ARMA 10)	Linear (ARMA 10)	90.97	-32	92.14	-42

TABLE 6.13: *Master test set with optimised insertion penalties (IP): effect of converting test data to the same trajectory-based features using the phone alignments of the original recognition system.*

The results of these experiments are listed in Tables 6.12 and 6.13. As stated above, the trajectory-based (Linear) systems were trained and this time the same type of test features could be evaluated (as stated in the "Test data" columns). As before, the values in the top part of Table 6.12 relate to the master test data with insertion penalties optimised on the development tests (following as the second set of results). Converting to the trajectory-based front-end features provided an accuracy close to the control (about 1% absolute difference for the linear system compared to the values reported in Table 6.3 on the master test data).

The optimised results of the master test set (Table 6.13), performed for more orders of ARMA filtering, indicate that the best results seem to be around a smoothness of order 6. This supports the finding in Section 6.5 that was obtained for trajectory-based systems built from features of higher smoothness. An ARMA filter order of 6 seemed to be adequate for good alignment.

In practice, it might be preferable to use only a single set of acoustic models (not requiring a standard phone recogniser to generate alignments) when performing the recognition of unseen test data. Phone alignments for the test features would then be generated, given a first pass of recognition, as was done for the systems listed in Table 6.10. We tested the viability of this approach (using only the trajectory-based features

System	Test data	ACC	IP	ACC (semi-tied)	IP
Linear	Linear	83.81	-46	86.05	-44
Linear (ARMA 6)	Linear (ARMA 6)	89.20	-55	90.16	-49
Linear (ARMA 7)	Linear (ARMA 7)	89.27	-48	90.36	-45
Linear (ARMA 8)	Linear (ARMA 8)	89.27	-48	90.59	-49
Linear (ARMA 10)	Linear (ARMA 10)	88.64	-42	89.88	-42

TABLE 6.14: *Master test set with optimised insertion penalties (IP): converting the test data to the same trajectory-based features using the phone alignments of first-pass recognition and performing a sweep of ARMA filter orders.*

for acoustic model estimation) and first performed a sweep of ARMA filter orders on the master test data (values appear in Table 6.14). The best accuracy was obtained for the system employing ARMA filtering of order 8. The results were verified for the development test data and insertion penalties shown in Table 6.15, which gave a slight improvement in accuracy (90.44%) compared to the accuracy for merely smoothing the test features (89.99%), shown in Table 6.10.

System	Test data	ACC	IP	ACC (semi-tied)	IP
Linear	Linear	83.77	-45	85.79	-31
Linear (ARMA 6)	Linear (ARMA 6)	89.02	-40	90.14	-48
Linear (ARMA 8)	Linear (ARMA 8)	89.22	-45	90.44	-38
Dev: Linear	Linear	83.46	-45	85.65	-31
Dev: Linear (ARMA 6)	Linear (ARMA 6)	87.76	-40	89.10	-48
Dev: Linear (ARMA 8)	Linear (ARMA 8)	88.13	-45	89.34	-38

TABLE 6.15: *Converting the test data to the same trajectory-based features, using the phone alignments of first-pass recognition and setting insertion penalties on the development set (Dev). A slight improvement in accuracy (90.44%) is achieved compared to the accuracy for merely smoothing the test features (89.99%) shown in Table 6.10.*

6.7 Conclusion

This chapter discussed the finding that the frame-based features of the filter bank that were closer to the spectrum fitted the piecewise linear description of phone transitions far better. In addition, ARMA filtering proved to be a beneficial pre-processing step for trajectory estimation. Recognition accuracy proved to remain stable at a tuned degree of smoothness. After including these optimisations in the approach, Chapter 7 explains how we continued to analyse the phone transitions in greater detail.

Chapter 7

Trajectory model optimisation

7.1 Introduction

Chapter 6 describes the choice of features for trajectory modelling at the spectral level. We confirmed, by obtaining reasonable recognition accuracy when using the modelled trajectories, that the modelling parameters selected were a reasonable approximation of the underlying features. The aim of this chapter is to describe a more detailed investigation of the modelling technique (not of the features modelled) to find and correct any remaining large approximation errors that remain. In general, the acoustic quality of diphone transitions varies widely. Therefore, to approximate these transitions better, we expected to make transition-specific model adjustments.

Section 7.3 describes how we continued to refine the model approximation metrics. Improved model approximation for a training data set does not on its own imply improved feature representation. Therefore, the ability of a trajectory model to predict a set of unseen test data accurately was also evaluated (Section 7.3.2).

Chapter 5 explains that the joining of transition models describes the trajectory of a whole utterance. A set of characteristic parameters make up the individual transition models. Breaking up an utterance model into smaller parts, up to a point where individual characteristic parameters describe different aspects of a single channel transition, allows detailed intra-transitional analysis. Since we could use these elementary parts to reconstruct any phone transition, the ability to predict characteristic parameters was of particular interest.

Before studying any improvement to the trajectory models, we performed a set of baseline experiments, described in Section 7.4. We focused in each of the next sections on a

single adjustment to the modelling approach and compared (1) the model approximation and (2) the predictive ability of the model with those of the baseline model.

7.2 Terminology

The few additional terms needed to describe certain detailed aspects of phone transitions better are listed in Table 7.1.

Term	Interpretation
Channel transition	A single feature channel of a single phone transition.
Predictor	An estimated value that predicts the value of a parameter to a degree of accuracy.
Mean value predictor	Using the mean value (estimated over a number of samples) as a predictor.
Reference stable value predictor	A predictor estimated for a stable value of a specific context.

TABLE 7.1: *Terminology used throughout this work.*

7.3 Model evaluation

This section introduces all the metrics necessary to compare the baseline model with the different model adjustments made to the trajectory models. Earlier chapters already include several model approximation metrics. Next, we reused and extended these metrics to work for the spectral models we intended to optimise. At the spectral level, a single trajectory (utterance model) modelled each feature channel of an utterance. Evaluating the difference between such an utterance model and the feature values directly did work, but was restrictive in the sense that the modelling choices could only be compared on the basis of a whole utterance, not for smaller segments.

It is well known that the phone context introduces significant variation, therefore the trajectory models should also be adjusted to fit the different types of phone transitions well. By further dividing the utterance into diphone transitions and then considering the trajectories of each diphone transition type, one could approximate a single characteristic change for each channel transition (element) of the diphone unit. (Section 5.3.2 explains how we previously performed a similar level of analysis for the cepstral models.) Now a refined metric was used (as defined in Section 7.3.1).

Normalising each feature channel's variance enabled comparisons, using model error. We compared the model errors with those in the previous chapter. Comparisons could be made between smaller elements in this way, such as comparing the individual channel transitions. Furthermore, grouping the characteristic (channel transition) models was

useful. In some of the experiments described later, we analysed the transitions pooled with regard to broad phone transitional classes and on a per-channel basis.

Referring to Section 5.3, we modelled a single feature channel of a diphone segment by means of a piecewise linear model. Essentially, this model consisted of three regions of interest: two stable value parts (roughly corresponding to phone centres) and the parts in between (change descriptors), describing the phone transition. We captured all the information to build any channel transition with these sub-components.

Section 7.3.2 then explains how we used key characteristic parameters to construct test trajectories. These test trajectories made it possible to compare different models with the test data. Models with parameters that were better predictors of unseen test data implied better understanding of the speech features analysed.

7.3.1 Approximating the training data

To evaluate the mismatch between the measured and modelled features, we used the two global metrics defined earlier, as well as two new diphone-specific variants of these metrics. The global metrics each provided a single value as an indication of overall model error, whereas the diphone-specific variants allowed us to analyse and compare the model error for different types of diphones.

The remainder of this section discusses the re-use of a number of the terms defined in earlier chapters (Sections 5.5.1.2, 5.3.1 and 3.3.2), specifically:

- The sample standard deviation of parameter p across all S samples within the set ω:

$$\sigma(p,\omega) = \sqrt{\frac{1}{S}\sum_{s=1}^{S}(p_s - \mu(p,\omega))^2} \tag{5.6}$$

- The MSE between the actual (p) and modelled ($\hat{p}$) parameter values measured across all S samples within the set ω:

$$MSE(p,\hat{p},\omega) = \frac{1}{S}\sum_{s=1}^{S}(p_s - \hat{p}_s)^2 \tag{5.2}$$

- The sample Pearson correlation coefficient between the actual (p) and modelled ($\hat{p}$) parameter values measured across all S samples with the set ω:

$$r(p,\hat{p},\omega) = \frac{\sum_{s=1}^{S}(p_s - \mu(p))(\hat{p}_s - \mu(\hat{p}))}{\sqrt{\sum_{s=1}^{S}(p_s - \mu(p))^2}\sqrt{\sum_{s=1}^{S}(\hat{p}_s - \mu(\hat{p}))^2}} \tag{3.1}$$

7.3.1.1 Global metrics

We review the two global metrics employed by using the above definitions. The first global metric was the global variance-weighted MSE calculated per channel ($GWMSE_{channel}$) from equation 6.5:

$$GWMSE_{channel} = \frac{1}{C}\sum_{c=1}^{C}\frac{MSE(p,\hat{p},\omega_c)}{\sigma^2(p,\omega_c)} \text{ where } \omega_c = \bigcup_{f=1}^{F_c}(p_{fc},\hat{p}_{fc}) \tag{7.1}$$

with C the number of channels and F_c the number of frames observed in channel c. (F_c is typically the same for all channels).

As the second global metric, we used the global correlation value ($Gr_{channel}$) from equation 6.7:

$$Gr_{channel} = \frac{1}{C}\sum_{c=1}^{C} r(p,\hat{p},\omega_c) \text{ where } \omega_c = \bigcup_{f=1}^{F_c}(p_{fc},\hat{p}_{fc}) \tag{7.2}$$

These two metrics produced similar trends in an earlier analysis (see Section 6.3.4). We report both to confirm our new observations.

7.3.1.2 Diphone-specific metrics

The earlier diphone-specific metrics used as described in Section 5.3.1 were not weighted for variance. The present chapter contains a definition of a variance-weighted diphone-specific metric, analogous to the global metrics above.

Specifically, we defined a variance-weighted diphone MSE for a particular diphone transition label d, as

$$WMSE_{diphone}(d) = \frac{1}{\sum_{s=1}^{S} CF_s}\sum_{s=1}^{S_d}\sum_{c=1}^{C}\sum_{f=1}^{F_s}\frac{SE(p_f,\hat{p}_f,\omega_{cd})}{\sigma^2(p,\omega_c)}$$
$$\text{where } \omega_c = \bigcup_{f=1}^{F_c}(p_{fc},\hat{p}_{fc}) \text{ and } \omega_{cd} = \bigcup_{s=1}^{S_d}\bigcup_{f=1}^{F_s}(p_{fc},\hat{p}_{fc}) \tag{7.3}$$

with S_d the number of samples of diphone d observed, and F_s the number of frames in sample s. Note that the variance used during normalisation was the channel-specific one, and was not dependent on the diphone identity. Similarly, one could use the Pearson correlation metric ($r_{diphone}$) on this level as well.

$$r_{diphone}(d) \quad = \quad \frac{1}{C}\sum_{c=1}^{C} r(p,\hat{p},\omega_{cd}) \text{ where } \omega_{cd} = \bigcup_{s=1}^{S_d}\bigcup_{f=1}^{F_s}(p_{fc},\hat{p}_{fc}) \tag{7.4}$$

Also, global MSE and correlation values can be calculated for a data set. For the variance-weighted diphone MSE this is defined as:

$$GWMSE_{diphone} \quad = \quad \frac{1}{D}\sum_{d=1}^{D} WMSE_{diphone}(d) \tag{7.5}$$

summed across the individual diphone labels d of the data set. As for the global metrics, we confirmed new observations for the trends of these measures across all the different transitions of the data sets.

7.3.1.3 Transition-specific metrics

Finally, channel transitions can also be analysed as separate classes. In this way, we investigated the individual feature channels of specific diphone clusters. Specifically to incorporate variance-weighting, we adjusted equation 7.3 to function for each channel separately ($WMSE_{trans}$):

$$WMSE_{trans}(c,d) = \frac{MSE(p,\hat{p},\omega_{cd})}{\sigma^2(p,\omega_c)}$$
$$\text{where } \omega_{cd} = \bigcup_{s=1}^{S_d}\bigcup_{f=1}^{F_s}(p_{fc},\hat{p}_{fc}) \text{ and } \omega_c = \bigcup_{f=1}^{F_c}(p_{fc},\hat{p}_{fc}) \tag{7.6}$$

As before, S_d represented the number of samples of diphone d observed, and F_s the number of frames in sample s. Section 7.5 explains how we used this measure to evaluate broad transitional phone classes and confirmed the effect of the experiments on model-feature mismatch for the individual feature channels of these classes.

7.3.2 Predicting the test data

Test data trajectories could be predicted by using the trajectory estimators that had been trained on the training data. By predicting the test data, we could choose the best

trajectory model to represent the speech features. Subdividing the modelled channel transitions into their constituent model parts made it possible to break up (and later reconstruct) the trajectories of a segment. For a single transition, we divided these parameters into two main categories: stable values and model alignments. As shown above, it was possible build a complete channel transition from a total of five parameters, namely $S1$, $S2$, T_{mid}, T_{dur} and Seg_{dur} respectively. Sections 5.5.4 and 5.5.5 explain that the model alignments ($T1$ and $T2$) in Figure 5.2 were derived directly from T_{mid} and T_{dur}. Test data trajectories can be predicted from the characteristic parameters (of a train data set). Therefore, estimating the distribution for each parameter of a channel transition allowed a compact representation of the train data trajectories.

In essence, an accurate description of the differences between phone transitions must be able to separate the phone classes. Correcting the approximation errors of a train data set might not always lead to improved class separation. It is imperative also to determine whether an adjustment to the model would improve the predictability of an unseen test data set.

To predict such a set of test trajectories, we first estimated the reference stable value predictors, given the training data. As this chapter explains, all that was necessary to accomplish this task was a mean value predictor for each stable value parameter ($S1$ and $S2$) of every unique transition. (Note that in this chapter we used a biphone context size for the estimation of each stable value predictor.) We could generate a complete test trajectory using ASR alignments and these stable value predictors. Initially, the ASR alignments were "oracle" alignments, but later we used a phone recognition system to align the test data. The ASR alignments provided a Seg_{dur} parameter value for each transition. Finding $T1$ and $T2$ became a matter of fitting a trajectory to a test segment (thus relying on the model structure), which provided complete model-feature mapping. This trajectory consisted of the mean stable value predictors ($S1$ and $S2$) and a change descriptor between $T1$ and $T2$. It followed that, once again (similar to Section 7.3.1), we could then evaluate trajectories by using the $GWMSE_{channel}$ and the $Gr_{channel}$ metrics. Measuring the difference between these trajectories and the features of the test data set enabled the use of a 3-piece modelling structure to find the trajectory models that were good predictors of test trajectories.

7.3.3 Experimental setup

This section specifies the trajectory model configuration and evaluation chosen for experimentation in all later sections. Specifically, all the experiments described in this

chapter focused on single adjustments to the trajectory modelling approach and compared the newly created models with previous model evaluation results (see Section 7.1). Given the findings in the previous chapter, we chose specific parameters and data sets to ensure that such comparable evaluation would be possible. Section 7.3.3.1 provides these details. We also explain the procedure for creating the trajectories for these data sets in Section 7.3.3.2. To provide more clarity on the general approach to comparing the evaluation results, Section 7.3.3.3 describes an experimental procedure.

7.3.3.1 Experimental data

When conducting the experiments discussed in this chapter, we continued to use the ATTC as a source of phone transitions. The specifics of this single speaker corpus had already been defined (see Section 4.2) so we only reviewed the features we used in the experiments. We generated the features according to the first four steps mentioned in Section 6.4.1, using the parameters optimised in Sections 6.4.2 - 6.5. This configuration was applied as follows:

- Use 26 filter bank channels.
- Sample each channel at a granularity of 5 ms per frame.
- Convert the raw filter bank outputs to log filter bank outputs.
- Subtract the mean of each feature channel on a per-utterance basis.
- Apply frame-based (low-pass) filtering, using an ARMA filter with a window size of six frames.
- Use a piecewise linear trajectory model with three pieces per diphone transition to model all the transitions of a complete utterance.

A segmentation process has to be performed to derive any trajectory model, which divides the frames of an utterance into diphone transitions (segments). (By concatenating the segment-based units, we estimated the trajectories at utterance level.) Chapter 6 explains how we used a set of oracle alignments (see Section 6.1), generated from the forced alignment of all available data. For the experiments discussed in this chapter, we used these same oracle alignments when we evaluate trajectory models. Using oracle alignments was a valid approach, since the goal was to select the best trajectory model based on data segments with known phone identities. We expected that the results generated with an "oracle segmentation" would be optimistic, since this approach used segments based on the known content of the test data. Therefore, we performed our

later analysis (in Section 7.9) by replacing all oracle alignments with blind recognition. We only used the oracle alignments during the initial analysis, as they provided an effective way of analysing the trajectories without the variance that possible errors might introduce.

Determining how well the trajectory models could predict feature trajectories required model evaluation on unseen test data. We used the standard master train and test data sets selected as described in Section 4.3 for this purpose. We analysed the model approximation for both these data sets and used the train data to predict the trajectories that we evaluated with the test data set.

7.3.3.2 Utterance models

ASR alignments (see Section 7.3.3.1) segment the utterances of the data sets into diphone segments. Essentially, every segment consists of many feature channels (26 in the current study). Constructing an utterance model required us to estimate a complete trajectory for each of these channels, spanning all the diphone segments of an utterance. Chapter 5, Section 5.3.1 introduces a transition model to model each channel of a diphone transition. Section 5.4.1 explains how to connect all the transition models for each feature channel, creating an utterance model. To this end, we used the "ConnectSegments" algorithm directly. Appendix B, Algorithm 5 provides a detailed definition of this procedure. Given a set of model alignments (spanning a whole utterance), the "ConnectSegments" algorithm finds a complete trajectory, using the least-squares fit for all line pieces of the piecewise linear model.

We defined more than one algorithm to find the model alignments for each segment, though a simple segment-based alignment algorithm (first creating all transition models) is adequate for most experiments and has the advantage of speed of execution. (Since the "AlignSegment" algorithm was common to almost all of the experiments described in this chapter, we introduced other algorithms only at a later stage.) The "AlignSegment" algorithm (see Algorithm 1 in Appendix B) fitted a transition model to the feature values of a single segment. As before, this process resulted in a set of model alignments.

7.3.3.3 Experimental procedure

By repeatedly comparing a set of highly specific measures on the same speech data sets, one could compare the experiments in the different sections that follow. Accordingly, this section describes in detail the process of trajectory estimation and measurement common to the experiments in this chapter.

For the purposes of this study, we analysed the model with each experiment by providing three main results, namely:

1. Model-feature approximation of the train and test data.
2. Model-feature approximation of specific diphones (as required).
3. Accuracy of trajectories' prediction of the unseen test data.

We generated utterance models for all (the train and test) data so as to estimate model-feature approximation in (1). As needed, we also continued to analyse the model-feature approximation of complete diphones in (2). Adequate approximation meant that the trajectories were representative of the features that were modelled and that we could proceed to generate the result in (3).

Once utterance models for a data set exist, the model-feature approximation can be evaluated. Estimating both the $GWMSE_{channel}$ and $Gr_{channel}$ metrics would provide these results. Alternatively, the diphone-specific metric $GWMSE_{diphone}$ could estimate the model-feature approximation of specific diphones. Finally, the model-feature approximation of trajectories that predict unseen test data had to be evaluated. Technically (3) in Section 7.3.3.3 differs from (1) only by the set of trajectory models used. The same $GWMSE_{channel}$ and $Gr_{channel}$ metrics were used.

We needed reference stable values (see Section 7.3.2) to construct trajectories that predicted the test data. Given the train data set, applying the "CalculateStableReferences" algorithm would find these values. Algorithm 6 details the procedure for processing through the utterance models and extracting stable values. Using this set of reference stable values as input into the "PredictedUtt" algorithm then generated the set of predicted trajectories (Algorithm 7 in Appendix B).

7.4 Baseline analysis

The first set of experiments established a baseline result. Section 7.4.1 explains how we assessed the ability of the models to predict the test data. We could then directly compare all the phone transition-based modelling improvements noted in the rest of this chapter with this baseline result. Further analysis of the model-feature approximation of specific diphones (in Section 7.4.2) made it possible to determine whether the modelling approach worked adequately for all the required phone transitions. The results given in Section 7.4.3 were then a more detailed analysis of specific phone transitions with the largest remaining error.

7.4.1 Predicting stable values for the test data

We trained trajectory estimators by using the stable values of a separate training data set (see Section 7.3.2). Accordingly, the accuracy with which features of an unseen test data set could be predicted depended on the model fit of the free trajectories for the training data. Next we continued by reporting first on the model fit between the free trajectories of the train data and test data. After this, it was possible to perform an analysis as a baseline to evaluate the remaining model-feature mismatch for the trajectories that predicted the features of unseen test data.

Model	$GWMSE_{channel}$	$Gr_{channel}$
Master train data		
3-piece	0.0577 (0.1615)	0.9707
3-piece (ARMA 6)	0.0160 (0.0557)	0.9920
Master test data		
3-piece	0.0578 (0.1587)	0.9706
3-piece (ARMA 6)	0.0160 (0.0541)	0.9919

TABLE 7.2: *Model fit between the free trajectories and the feature values of the master train data and the test data sets clearly shows a benefit for activating the ARMA filtering option.*

Table 7.2 lists the $GWMSE_{channel}$ and $Gr_{channel}$ measurements explained in Section 7.3.1.1. (The mean of the standard deviation across the 26 channels is provided in brackets.) We used two options: the one a plain 3-piece utterance model and the second an option including ARMA filtering (Model) to represent the speech data. In both cases, the results obtained for the master train data and test data sets were included. The benefit of the ARMA filtering option for model fit was clearly visible, once again. Moreover, the master test data and train data sets showed a closely similar model fit. Given these findings, we expected to note an improved stable value estimation for ARMA filtering.

Model	$GWMSE_{channel}$	$Gr_{channel}$
Master test data		
3-piece	0.1669 (0.3402)	0.9130
3-piece (ARMA 6)	0.1180 (0.2234)	0.9397

TABLE 7.3: *Mismatch between the baseline-predicted trajectory options and the actual feature values of the test data set. ARMA filtering improves the predictability of the frames of the 3-piece utterance models.*

Next we measured the remaining model-feature mismatch for the trajectories that predicted the features of the unseen test data set. As discussed, constructing these trajectories only required a set of seen stable value class estimates of the training data. In these experiments, we also continued to evaluate 3-piece utterance models, with and without ARMA filtering (Model). Table 7.3 shows the values for the $GWMSE_{channel}$

and the $Gr_{channel}$ metrics. By strengthening the results given in the previous chapter this made it clear that ARMA filtering improved the predictability of the frames of the 3-piece utterance models.

7.4.2 Analysing phone transition error

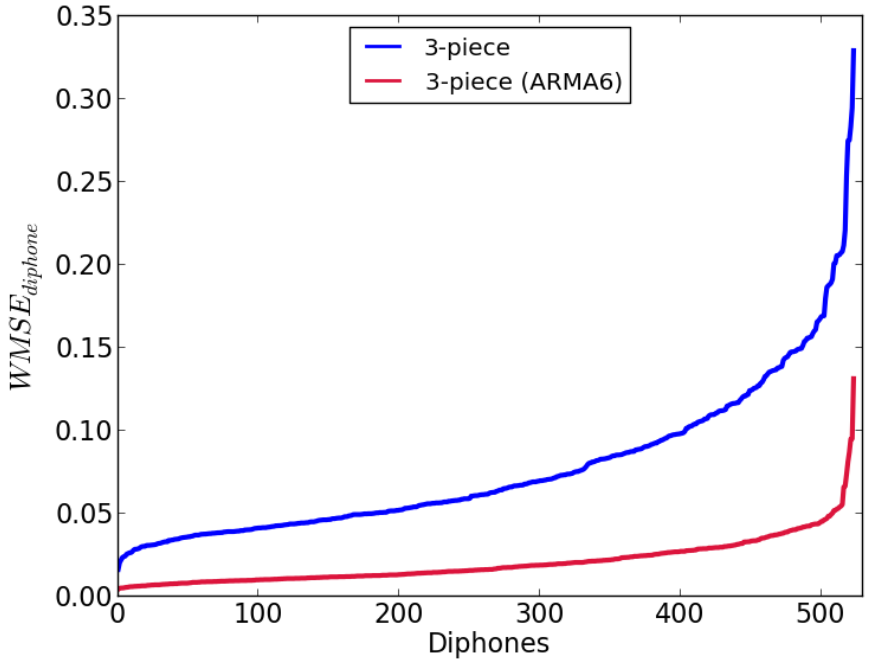

FIGURE 7.1: *Master train data: model fit of free trajectories for diphone transitions using the $WMSE_{diphone}$ metric, ranking these error values to reveal the number of more problematic transitions.*

Equations 7.3 and 7.4 are particularly useful to describe model fit for specific diphone transition labels (across multiple examples). For each diphone transition d of the master train data set, we estimated such $WMSE_{diphone}$ and $r_{diphone}$ values. Diphones have widely differing characteristics, which should significantly affect how the feature values of a particular diphone class behave in general. For this reason, ranking these error values and investigating the larger errors might reveal a number of more problematic transitions, where the model under study could be improved. To this end, Figures 7.1 and 7.2 compare graphs of the same 3-piece linear segment model approximation options of Table 7.2. We plotted the result of metrics $WMSE_{diphone}$ and $r_{diphone}$ for the master train data set (as do Figures 7.3 and 7.4 for the test data segments).

Using 3-piece models directly on the log filter bank outputs confirmed larger errors (for all diphones), than for smoothed features (ARMA 6). Furthermore, both of the modelling options showed a tail of diphone transitions with a far larger error (lower

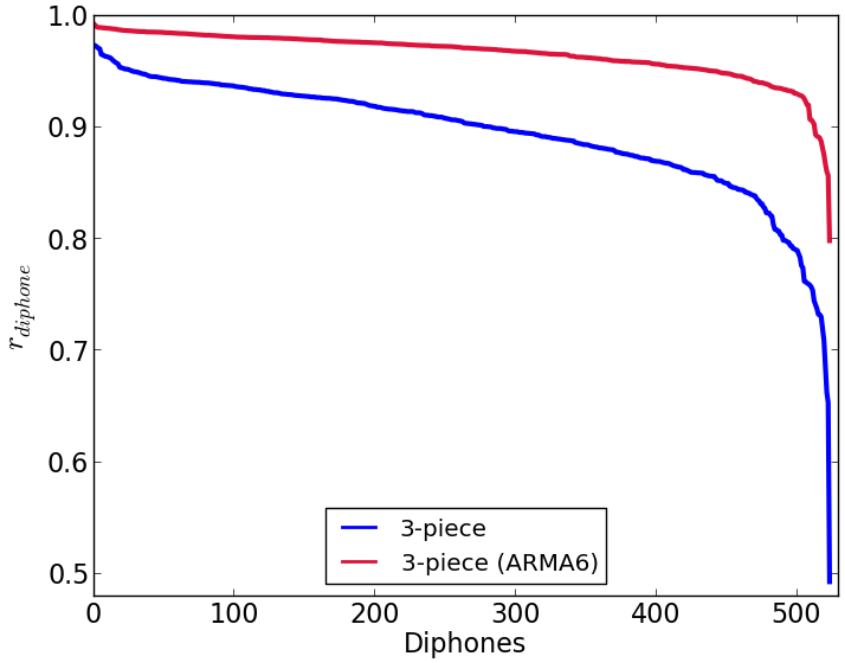

FIGURE 7.2: *Master train data: model fit of free trajectories for diphone transitions using the $r_{diphone}$ metric, ranking these error values to reveal the number of more problematic transitions.*

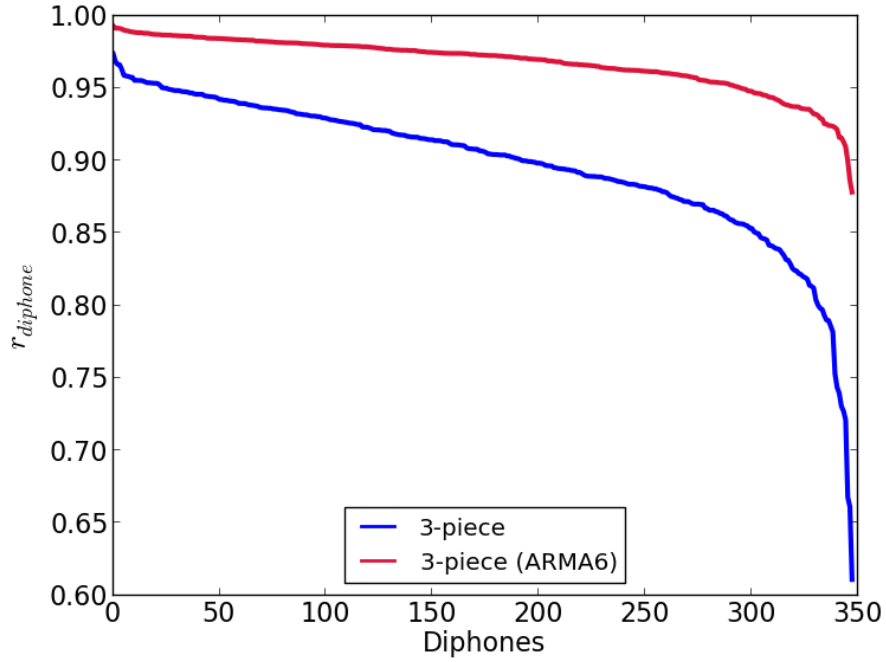

FIGURE 7.3: *Master test data: model fit of diphone transitions using the $r_{diphone}$ metric.*

correlation) than the mean for the same option. It is clear from the figure that frame-based smoothing reduces this effect (diphone error increases gradually with a more

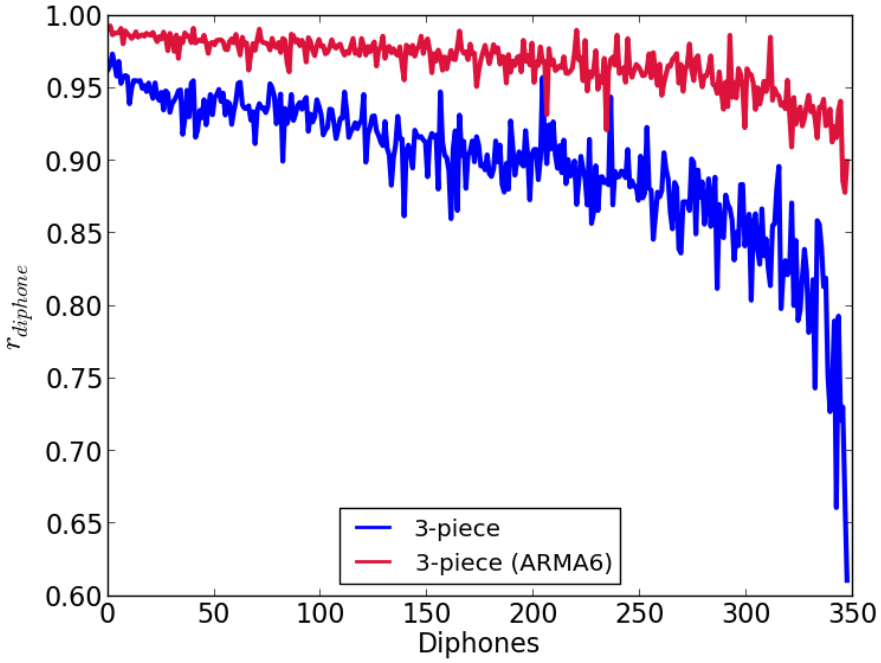

FIGURE 7.4: *Master test data: model fit of diphone transitions using the $r_{diphone}$ metric and using train set transition ordering.*

distinct tail and fewer transitions for the test data were part of this category). We chose to analyse only the trajectories with smoothed (ARMA 6) features after this point. Lastly, Figure 7.4 clearly shows that specific diphone transitions behave similarly in the train and the test data sets. Ranking the test data transitions to follow the same order as for the train data (Figure 7.2) produced graphs with a similar shape as those shown in Figure 7.3.

7.4.3 Transitions with large error

As indicated in the section above, a tail of diphone transitions showed a far larger error than the mean of all the diphone transition values. These diphone transition values are either $WMSE_{diphone}$ or $r_{diphone}$ values in Figures 7.1 to 7.4. Next we specifically investigated these transitions with a larger error in the training data, in an attempt to determine the cause of the error and decide whether further model improvements were needed. The severity of the observed error for the diphones of a particular option can be quantified. To this end, we estimated the mean of the $WMSE_{diphone}$ measurements (namely the $GWMSE_{diphone}$ value) and a corresponding standard deviation. Then we investigated diphones with an error value larger than two standard deviations from the $GWMSE_{diphone}$ value. For example, in the case of a 3-piece (ARMA 6) model, the

$GWMSE_{diphone}$ value is 0.0204 and its standard deviation (σ) is 0.0139. This means that 2σ higher than $GWMSE_{diphone}$ has a value of 0.0483.

Transition	$WMSE_{diphone}$	**# Examples**
b=@u	0.0484	45
k=2:	0.0484	21
O=t	0.0494	26
i=d	0.0502	84
i=i@	0.0518	26
i@=t	0.0519	111
d=9y	0.0527	35
k=i@	0.0530	27
@=@i	0.0539	17
t=u@	0.0544	77
N=u@	0.0554	10
@=A:	0.0659	71
k=d	0.0665	25
d=u@	0.0735	15
s=s	0.0812	54
u@=k	0.0870	40
@=u@	0.0949	48
@=9y	0.0951	40
@=i@	0.1313	11

TABLE 7.4: *Diphone transitions with high $WMSE_{diphone}$ error (at least 2σ from the $GWMSE_{diphone}$ value) for 3-piece (ARMA 6) models.*

Transition	**# within word**	**# word boundary**	**Fraction (word boundary)**
@=A:	0	71	1.000
s=s	0	54	1.000
@=9y	0	40	1.000
@=i@	0	11	1.000
k=d	2	23	0.920
N=u@	1	9	0.900
@=u@	8	40	0.833
@=@i	3	14	0.824
t=u@	64	13	0.169
i=d	79	5	0.060
i=i@	25	1	0.038
k=i@	26	1	0.037
u@=k	39	1	0.025
i@=t	110	1	0.009
b=@u	45	0	0.000
k=2:	21	0	0.000
O=t	26	0	0.000
d=9y	35	0	0.000
d=u@	15	0	0.000
Total	**499**	**284**	**0.363**
Fraction (train set)	**0.0067**	**0.0038**	

TABLE 7.5: *Word boundary analysis of diphone transitions with a high $WMSE_{diphone}$ error showing a limited set of "unexpected" diphone transition labels formed between words (top part of the table).*

In the 3-piece (ARMA 6) models, 19 diphones displayed errors larger than 2σ from the $GWMSE_{diphone}$ value. These transition labels (Transition) are listed in Table 7.4. Each

label contains the two phone identities involved in the diphone transition, where "=" separates each phone. The diphone transitions are ordered according to $WMSE_{diphone}$ errors and then the number of examples is provided in a last column entry (# Examples).

So far, the modelling technique did not distinguish between phone transitions within or between words (an utterance is fully described by the diphone unit sequence, without pauses). Therefore we checked where these problematic transitions occurred: were they on word boundaries (# word boundary) or also part of the pronunciations of words in the pronunciation dictionary (# within word)? Table 7.5 shows the counts of the number of transition examples for each of these categories. Finally, the "Fraction (word boundary)" heading in Table 7.5 provides the ratio of diphone transition examples on word boundaries compared to the within-word transitions.

Transition	**# within word**	**# word boundary**	$WMSE_{diphone}$
@=@	11	113	0.0213
s=s	0	54	0.0812
t=t	0	20	0.0183
x=x	0	19	0.0147
n=n	0	13	0.0060
k=k	0	13	0.0243
r=r	0	10	0.0062
f=f	2	9	0.0049
l=l	0	8	-
m=m	0	6	-
a=a	0	3	-
p=p	0	1	-
A:=A:	2	0	-
Total	**15**	**269**	

TABLE 7.6: *Number of examples for diphone labels consisting of repeated phones, showing that most of these transitions occur between words.*

It can be seen from the results listed that in total 499 diphone examples are due to within-word transitions, which is more than 63%. A limited set of "unexpected" diphone transition labels is formed between some words (top part of the table). At this point, the number of diphones exceeding the 2σ error boundary constituted only about 1% of the training data, a negligible number.

Repeated phones generated another group of diphone labels, which might have been exceptions instead of the rule. Since strictly speaking no phone transition took place, the model alignment estimation for these transitions might be inaccurate. Table 7.6 lists the number of transition labels in the training data that are part of this category. The counts were given to investigate whether these diphones formed because of word boundaries (# word boundary) or not (# within word). Then we calculated the MSE ($WMSE_{diphone}$) to show the goodness of model fit, and we only estimated this value for the transitions with more than 10 examples of training data.

As shown in Table 7.6 only one of the repeated phone transitions had a $WMSE_{diphone}$ value greater than the 2σ error boundary (/s=s/). All the other diphones had an MSE closer to the mean than a single standard deviation. Virtually all these labels were due to word boundary effects (only 15 labels were formed by dictionary words). (The repeated label /A:=A:/ is not a valid pronunciation in the Afrikaans language and in this case, this was due to inaccuracy in the G2P pronunciation modelling.)

Table 7.5 shows that all the diphone transitions (except /s=s/ and /@=A:/) include a phone from two specific broad phone categories: stops or diphthongs. From a linguistic perspective, we expected the phones of these categories to include greater complexity. Stop sounds produce short in-phone transitions because of aspiration (airflow burst after the release of a closure). By contrast, diphthongs are long vowel sounds that gradually change in acoustic quality, also leading to extra transitions within a phone.

To investigate the effect of these diphones for trajectory modelling, we started by performing a few spot checks on the phone transitions with large error. Figure 7.5 shows an example of a plot to compare ten transitions of the 12th feature channel for /t=u@/. The examples have slightly different durations (in terms of frames) and 10 ASR boundaries. We found the mean ASR boundary (at 20 frames relative to a segment start index of 0) and then shifted the plot of each channel transition so that every ASR boundary was at 20 frames.

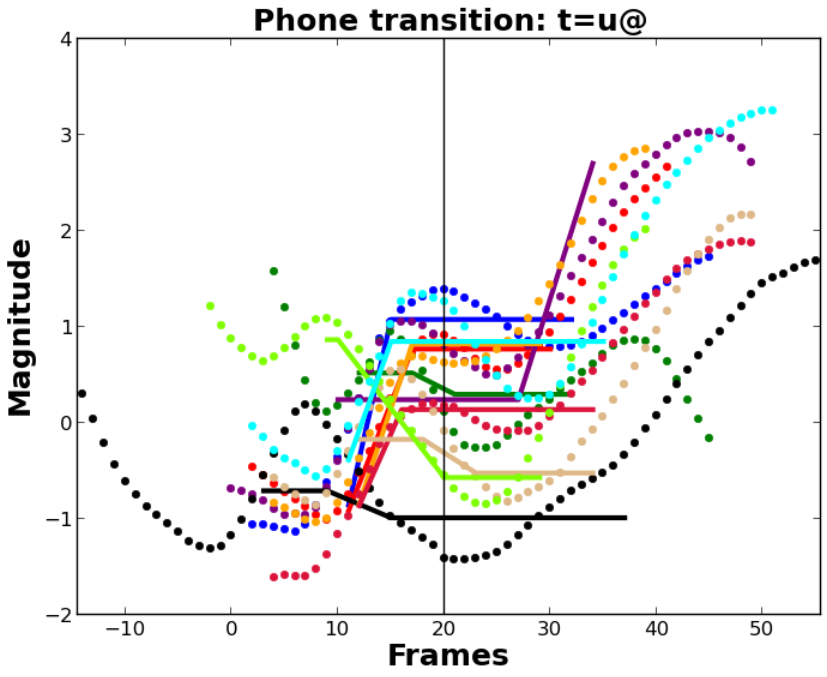

FIGURE 7.5: *Ten transitions (randomly selected from the master test set) of the 12th feature channel for the diphone transition /t=u@/, aligned with regard to the ASR boundary (black vertical line).*

Double transitions are clearly visible for the feature values (the dots in the figure) of this channel transition. When overlaying the transition model representation (modelling only the frames belonging to this diphone), it is clear that the 3-piece linear model is not a good representation of these double transitions. Specifically, the estimation of the stable value parts is severely affected when change detection fails (such as the black trajectory, where the model detects almost no change).

We investigated three possible causes of transition errors in the models: (1) word boundary effects, (2) repeated phones and (3) double transitions. The findings indicate that word boundary effects only affect a small number of trajectories. Likewise, repeated phones cause little error, but double transitions require further investigation. The next section explains how we chose to correct these errors, catering for possible double transitions occurring with the broad phone classes of diphthongs and stops.

7.5 Splitting phones

This section corrects the inability of a 3-piece model to track the double transitions seen in diphones of the diphthong and stop broad phone classes. The resulting lower model error should increase the accuracy with which trajectory models predict unseen test data. Previously, as discussed in Chapter 5, we used transition models with more elaborate change descriptors (4-piece segments) to accommodate the intricate changes of segments at the cepstral level. Since the feature channel changes of the spectral level correlated more directly with the observed changes in phone quality (at a particular time instant), another approach would be simply to split such segments. Each of the trajectories for these diphones would then be constructed of two 3-piece segments, allowing more model parameters to describe these transitions.

Creating two diphones for each phone transition where one monophone belongs to the "stop" or "diphthong" broad phone classes requires a new set of phone alignments to segment the speech. We used exactly the same system as described in Section 6.3.3 for this purpose. This process only required another set of input pronunciations. For example, it was now possible to represent the phone /t/ as the two phones /t1/ and /t2/, which then generated the desired set of forced alignments.

7.5.1 The effect of splitting segments

To investigate the overall effect of splitting segments, Table 7.7 compares the global model approximation (in terms of $GWMSE_{channel}$ and $Gr_{channel}$ metrics) with the

Model	$GWMSE_{channel}$	$Gr_{channel}$
Master train data		
3-piece (ARMA 6)	0.0160 (0.0557)	0.9920
Split phones	0.0099 (0.0378)	0.9950
Master test data		
3-piece (ARMA 6)	0.0160 (0.0541)	0.9919
Split phones	0.0097 (0.0361)	0.9951

TABLE 7.7: *Comparing the model fit between the free trajectories and the feature values of the master train data and test data sets show that splitting segments improve model approximation.*

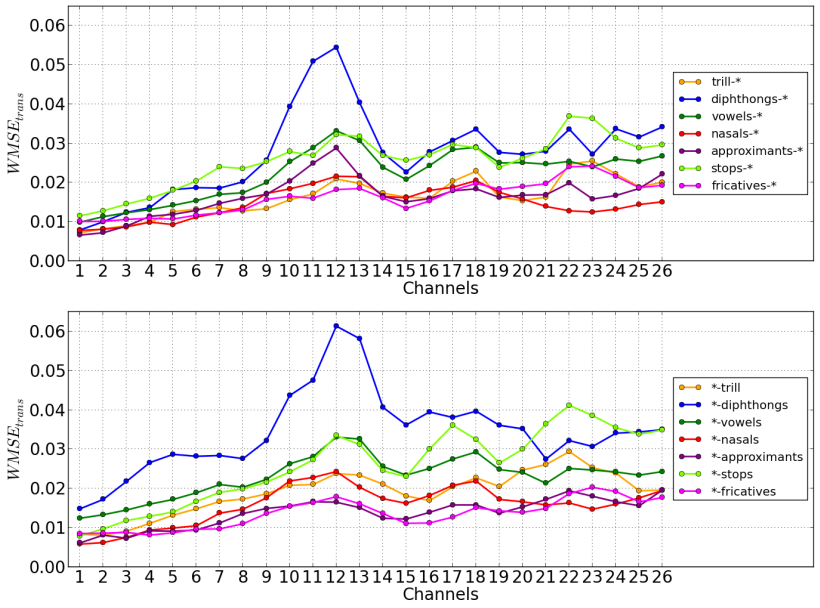

FIGURE 7.6: *Analysis of the training data before splitting phones, for each broad transitional phone class d and feature channel c with the $WMSE_{trans}$ measure. Both broad transitional classes "to" (bottom) or "from" (top) of phones belonging to a particular broad phone transitional class are examined. Phones belonging to the diphthong class resulted in far larger error than other phone transitional classes.*

previous values (3-piece (ARMA 6) in Table 7.2). For each segment we model the feature trajectories using 3-piece linear transitions and smooth the feature channels with an ARMA filter of the 6th order. According to the results, splitting segments does improve the feature model approximation further. To obtain a clearer picture of this effect, the three figures (Figure 7.6, 7.7 and 7.8) are particularly helpful. In these figures, we calculate the $WMSE_{trans}(c, d)$ metric on the master train and test data. This makes it possible to plot the result of each feature channel c and broad phone transitional class d and verify (after splitting segments) whether any broad transitional classes still present large errors.

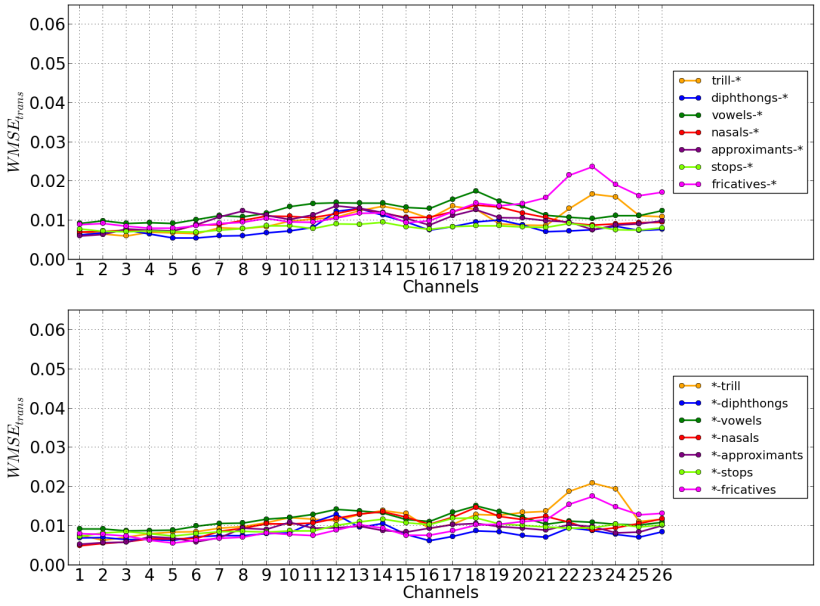

FIGURE 7.7: *Analysis of the training data after splitting phones, for each broad transitional phone class d and feature channel c with the measure $WMSE_{trans}$. Both broad transitional classes "to" (bottom) or "from" (top) of phones belonging to a particular broad phone transitional class are examined. The different classes of phone transitions behave much more similarly across all the feature channels.*

In figure 7.6, one sees that both phone transitions "to" or "from" a phone belonging to the diphthong class resulted in far larger errors than for other broad phone transitional classes. In particular, channels 10–14 seemed to be more problematic. Overall, stops were clearly the second-most difficult broad transitional phone class to model, followed by vowels. In general, lower feature channel numbers also did slightly better than higher feature channel numbers (a finding that made sense, considering that lower channel numbers are associated with lower frequency components).

Splitting the phones of the diphthong and stop phone classes had a strong effect. Figures 7.7 and 7.8 show the same measure as before but now this is estimated for the new segments of both data sets. The different classes of phone transitions behave very similarly across all the feature channels. Transitions of the fricative class do show slightly more error for the higher channel numbers of the test data, which makes sense given the greater energy of high-frequency components of this class.

In order to verify and put the improvement into perspective, we estimated the $WMSE_{diphone}$ measure again for the new set of training segments. This repeated the experiment presented in Table 7.5 (Table 7.8). Then only three diphones (/@=A:/, /s=s/ and /s=a/) displayed an error greater than 2σ from the $GWMSE_{diphone}$ value in Section 7.4.2.

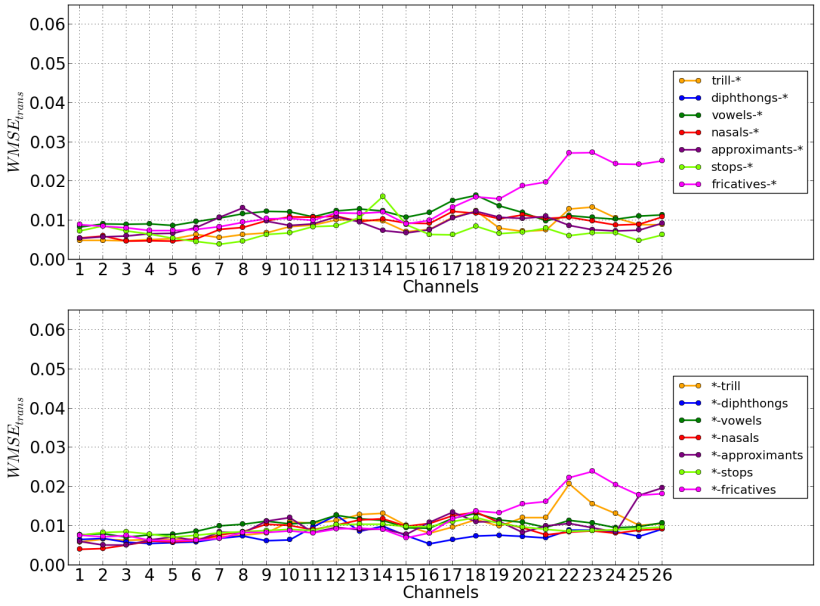

FIGURE 7.8: *Analysis of the test data after splitting phones, for each broad transitional phone class d and feature channel c with the measure $WMSE_{trans}$. The analysis confirms the similar behaviour of phone transitions for classes different from the training data.*

When phones have been split, the $GWMSE_{diphone}$ value becomes 0.0104 and its standard deviation (σ) is 0.0087. This means that 2σ from the mean is now far lower and has a value of 0.0277. Table 7.8 provides a list of the diphones with a $WMSE_{diphone}$ value greater than this new cut-off value. Fewer transitions form part of this list, a total of about 0.5% of all the diphones in the training data. (Note that the counts for the diphones /@=A:/ and /s=s/ in Table 7.8 differ from the numbers in Table 7.5, because the acoustic alignments improved, so that more short silences between word boundaries could be detected.) To check, we also reconsidered the repeated phone analysis for the diphones of the new data set shown in Table 7.9. This category remained fairly stable, and in a few places the error increased slightly.

7.5.2 Predicting stable values for test data

After splitting the segments, the train data and test data sets included a considerable number of new phone transitions. We evaluated the predictive ability of the trajectory model for unseen test data for these new train and test data sets. Moreover, we compared this result of the new test data set directly with the outcome of the previous experiment (Section 7.4.1), enabling us to choose the best representation.

Transition	# within word	# word boundary	Fraction (Word boundary)
@=A:	0	72	1.000
s=s	0	61	1.000
N=O	0	17	1.000
@=E	0	11	1.000
u=@	1	9	0.900
N=A:	2	14	0.875
@=@i1	3	15	0.833
@=a	25	86	0.775
t2=n	10	13	0.565
s=a	51	32	0.386
n=a	103	11	0.097
j=O	10	0	0.000
Total	**205**	**341**	**0.625**
Fraction (train set)	**0.0021**	**0.0035**	

TABLE 7.8: *Diphone transitions with high $WMSE_{diphone}$ error for 3-piece (ARMA 6) models after splitting the phones of the stops and diphthong classes showing fewer transitions than the previous result (Table 7.5) for a lower cut-off value.*

Transition	# within word	# word boundary	$WMSE_{diphone}$
@=@	11	113	0.0222
s=s	0	61	0.1446
x=x	0	19	0.0146
n=n	0	13	0.0035
r=r	0	11	0.0230
f=f	2	9	0.0045
l=l	0	9	-
m=m	0	6	-
a=a	0	3	-
A:=A:	2	0	-
Total	**15**	**244**	

TABLE 7.9: *After splitting phones: the number of examples seen for diphone labels consisting of repeated phones remains fairly similar to what was detected previously (Table 7.6).*

Model	$GWMSE_{channel}$	$Gr_{channel}$
Master test data		
3-piece (ARMA 6)	0.1180 (0.2234)	0.9397
Split phones	0.1143 (0.2160)	0.9414

TABLE 7.10: *Comparing the mismatch between predicted trajectories and the actual feature values of the test set for 3-piece (ARMA6) models with and without split phones. Splitting the segments produced more accurate test trajectories.*

Listing both of these results, Table 7.10 shows the model-feature mismatch of the master test data, evaluated in terms of the same $GWMSE_{channel}$ and $Gr_{channel}$ metrics. We found that splitting the segments produced more accurate test trajectories.

7.6 Improving pronunciations

We point out in Section 7.4.3 that a repeated phone transition label (/A:=A:/) occurs in the data set. Transition label /A:=A:/ is not a valid pronunciation (not even as a word boundary effect) in the Afrikaans language. The reason is that we do not obtain all the pronunciations for the words in the transcriptions of the ATTC (see Chapter 4) from pronunciation dictionaries. To avoid the problem (up to this point), we used the default&refine algorithm [95] to generate G2P-based pronunciations for all the dictionary words. These pronunciations were therefore predictions and were not expected to be perfect.

The G2P rules were estimated by using the *Closely Related Languages Afrikaans Pronunciation Dictionary* (RCRL APD) [96], which contains 32000 words. From these, 3762 words can be used as they are for the master train data and test data sets of the ATTC. The remaining 1 410 words still needed pronunciations. To improve the accuracy of the predicted pronunciations, we manually verified the G2P-based pronunciations for this list of words. Moreover, since many of these tokens contained spelling errors, relatively few audio examples could map directly to these words. The spelling errors, therefore, allowed the fine-tuning of the pronunciation for a token (introducing additional tokens), to match the audio more accurately. Completing this process changed the pronunciations of 287 words. We present the updated results for this new pronunciation dictionary in Section 7.6.1 below.

A second effect influencing the modelling of pronunciation (using trajectories) is that of corpus segmentation. Section 7.3.3.1 explains that oracle alignments are generated to segment the speech data into monophones. (From these monophone segments, we later constructed the diphone transition segments.) For this reason, we include a section on corpus segmentation (Section 7.6.2) which provides a discussion on the impact that the combination of segmentation and frame-based filtering had on trajectory model estimation.

7.6.1 Results with the new dictionary

To establish a baseline for the next experiments, we extended the results in Tables 7.7 and 7.10 with the model approximation and the predicted trajectory tracking of test data features for the new pronunciation dictionary. Splitting the segments according to the procedure described in the previous section produced new results, which are shown in Tables 7.11 and 7.12. The updated values for the $GWMSE_{channel}$ and $Gr_{channel}$ metrics are provided.

Model	$GWMSE_{channel}$	$Gr_{channel}$
Master train data		
Split phones (old dict)	0.0099 (0.0378)	0.9950
Split phones (new dict)	0.0100 (0.0382)	0.9950
Master test data		
Split phones (old dict)	0.0097 (0.0361)	0.9951
Split phones (new dict)	0.0098 (0.0364)	0.9951

TABLE 7.11: *Model fit between the free trajectories and the feature values of both the master train data and test data sets, using the improved pronunciation dictionary, do not show an effect on model approximation.*

The new dictionary did not have an effect on model approximation - compare the "Split phones (old dict)" values in Table 7.11 with the result of the new improved segment identities "Split phones (new dict)". The predicted test trajectories might be different. As with the split phones, the new dictionary gave rise to a new set of diphone transition examples. In order to verify the trajectory tracking of the predicted test data trajectories, the result shown in Table 7.12 was important. We found that these numbers remained close enough to state that globally the prediction accuracy also remained similar.

Model	$GWMSE_{channel}$	$Gr_{channel}$
Master test data		
Split phones (old dict)	0.1143 (0.2160)	0.9414
Split phones (new dict)	0.1152 (0.2170)	0.9409

TABLE 7.12: *Model tracking of the predicted test trajectories and the test features for the improved pronunciation dictionary show that global prediction accuracy remains similar.*

System	**ACC**	**IP**	**ACC (semi-tied)**	**IP**
Control (old dict)	93.06	-44	93.97	-35
Control (new dict)	93.40	-35	94.15	-35
Dev: Control (old dict)	91.93	-44	93.27	-35
Dev: Control (new dict)	92.22	-35	93.25	-35

TABLE 7.13: *Comparing phone recognition results for ARMA 6 systems trained by using the old and the updated pronunciation dictionaries indicates slightly higher recognition accuracy for the system trained with the new pronunciation dictionary.*

Lastly, we included a phone recognition result. These accuracies provided an update on the previous findings in Section 6.5.2 for the improved pronunciations of the new dictionary. As in those sections, we called the phone recognition system the "control", which had been trained on filtered (ARMA 6) features, but did not yet follow a trajectory model. Table 7.13 includes the phone recognition accuracies (ACC and ACC semi-tied) for both development set (Dev) and master test data. As before, we optimised the insertion penalty (IP), using the development set result. We found that the system

achieved slightly higher recognition accuracy for the control system trained with the new pronunciation dictionary.

7.6.2 Corpus segmentation

In the previous chapter, we explain how we used the segmentation system described in Section 6.4.2 to derive feature alignments for monophone units. In addition, chapter 6 shows that the phone accuracies for recognition systems employing ARMA filtered features of high order compare well with standard accuracies. In some cases, these systems even improved the phone accuracies compared to those trained with standard MFCC features. The question arises whether filtered features should rather be used to segment the utterances for trajectory modelling. To investigate this, we tested the effect of segmentation derived from filtered features on the model tracking of predicted test trajectories. This section first gives an overview of the results and then explains the alignment choices made after this stage.

In particular, we considered three scenarios:

1. No low-pass filtering of features for the segmentation system and no filtering when estimating trajectory models.
2. ARMA filtering of the 6th order for the segmentation system and for the estimation of trajectory models.
3. No filtering during segmentation, but ARMA 6 filtering of features for trajectory estimation.

Option	**Scenario**	**Details**	$GWMSE_{channel}$	$Gr_{channel}$
a	1	old dict	0.1669 (0.3402)	0.9130
b	2	new dict	0.1288 (0.2514)	0.9338
c	2	new dict, split phones	0.1256 (0.2430)	0.9355
d	3	old dict	0.1180 (0.2234)	0.9397
e	3	old dict, split phones	0.1143 (0.2160)	0.9414
f	3	new dict, split phones	0.1152 (0.2170)	0.9409

TABLE 7.14: *Model tracking of the predicted test trajectories and the test features showing the effect of corpus segmentation for trajectory modelling. Performing no filtering during segmentation, but ARMA filtering of features for trajectory estimation (scenario 3) is the better choice.*

Table 7.14 lists different sets of results for each of the above-mentioned scenarios (Option). More detailed descriptions (Details) for each result are shown to indicate which pronunciation dictionary was used and whether or not split phones were applied. Similar

to previous sections, $GWMSE_{channel}$ and $Gr_{channel}$ values are provided to describe the model tracking for the predicted trajectories of the test data.

The finding in Chapter 6 (that it is better to use ARMA to filter features before trajectory estimation) is obvious when comparing options (a) and (d) in Table 7.14. We achieved a far lower correlation ($Gr_{channel}$) when predicting test trajectories with the option (a) feature sets. However, ARMA filtering seemed to have a detrimental effect on alignments when using the ASR systems built from these features to segment speech data.

Compare options (c) and (f) to determine whether ARMA filtering should be used during segmentation. Better correlation was achieved when ARMA filtering was not used during segmentation. Given these findings, we continued to use scenario 3 from this stage onwards: no filtering during segmentation, but ARMA 6 filtering of features for trajectory estimation. (This is the same set-up as that described in Section 7.3.3.1).

7.7 Improving stable values on turning points

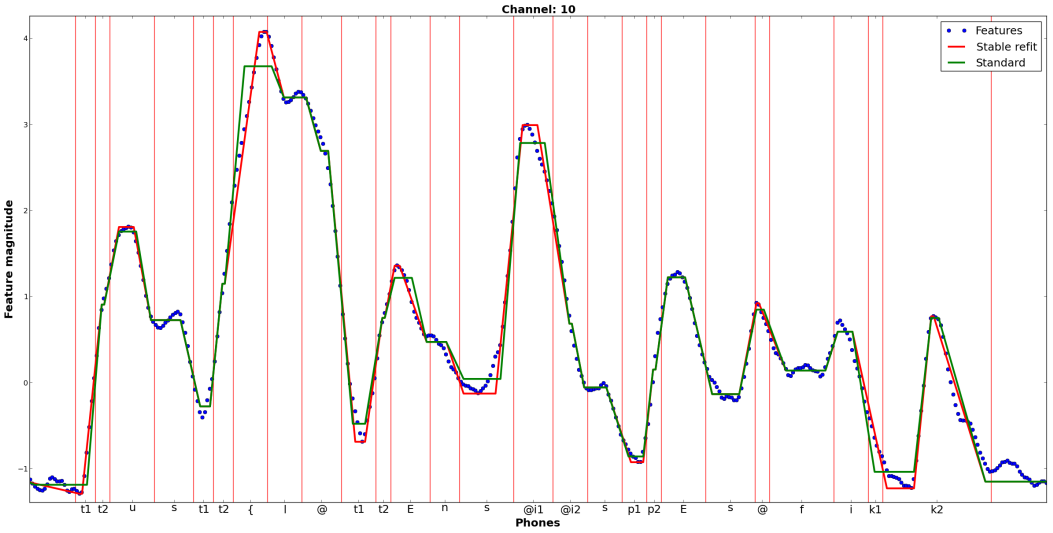

FIGURE 7.9: *Analysis of the free trajectory fit of a single feature channel, showing the 3-piece model fit (standard) and the improved 3-piece model fit (stable refit), where the fit of a stable value at turning points was improved.*

Figure 7.9 shows an example plot of the free trajectory (Standard) and feature values (Features) for the tenth channel of a single utterance. An examination of several randomly selected examples with similar plots revealed that many of the larger errors were found at the points in time where the feature values reached their local minimum or maximum.

Specifically, where feature values reached a local maximum or minimum, stable values were sometimes inaccurate. When feature values near adjacent change descriptors did not closely resemble a straight line, the current algorithm had no choice but to estimate the mean of rapidly changing features. This mean value then became a stable value. Figure 7.9 gives a clear example. For the first stable value ($S1$) of the diphone transition /{=l/, eight frames form a local maximum for the feature values at a significantly higher magnitude than the free trajectory.

Model	$GWMSE_{channel}$	$Gr_{channel}$
Master train data		
Split phones	0.0100 (0.0382)	0.9950
Split phones + Stable refit	0.0069 (0.0288)	0.9965
Master test data		
Split phones	0.0098 (0.0364)	0.9951
Split phones + Stable refit	0.0068 (0.0273)	0.9966

TABLE 7.15: *Model fit between the free trajectories and the feature values of the master train data and test data sets using the improved stable value fits show improved model approximation when applying the "RefitStable" algorithm.*

To improve stable value estimation, we performed a second pass of the final trajectory fit. For this purpose, we defined a new algorithm and called it "RefitStable". Only when we encountered a stable value where the slopes of the change descriptors it connected were of a different sign, did we shift the stable value to the maximum (or minimum) feature value. This new value lay within the local model alignments of the current stable value. The search for new local model alignments provided an optimal model fit where the new stable value then allowed us to refit a new trajectory. This algorithm is described in more detail in Appendix B. (see Algorithm 9.) Figure 7.9 shows this revised free trajectory as the red line (Stable refit).

Model	$GWMSE_{channel}$	$Gr_{channel}$
Master test data		
Split phones	0.1152 (0.2170)	0.9409
Split phones + Stable refit	0.1158 (0.2217)	0.9404

TABLE 7.16: *Effect on accuracy of predicted test trajectories when applying the "RefitStable" algorithm during training. Almost not change is detected.*

In the same way as described in the previous sections of this chapter, we then compared the global model approximation with the values of the preceding section. We provided the model approximation values in terms of the $GWMSE_{channel}$ and $Gr_{channel}$ metrics. The results shown in Table 7.15 indicate that the model approximation of free trajectories is improved when applying the "RefitStable" algorithm at a corpus level. For the train data and test data sets, we found that the $GWMSE_{channel}$ metric dropped by a value of 0.003. A better model approximation at this level had little effect on stable value predictors. Table 7.16 provides the updated results when predictors have been

trained on the improved trajectories. The trajectory tracking for predicted test data with this new set of predictors shows almost no change.

7.8 Trajectory tracking with dynamic programming

Free trajectory tracking in all previous sections of this chapter utilises the "AlignSegment" and "ConnectSegments" algorithms (Algorithms 1 and 5), in which we search for the model alignments ($T1$ and $T2$) of every diphone transition. These searches only take place for feature values on a per-segment basis. Since each diphone forms part of a larger context within an utterance, additional information about any particular transition can be derived from other diphone examples in the immediate vicinity. Therefore, only sharing stable values (when joining adjacent diphone segments to construct an utterance model) may not provide optimal model alignments for the local transitions. With this section, we show that a DP algorithm can track the trajectory of complete feature channels in an utterance while effectively including frames of adjacent diphones in the tracking process of a current diphone. This process then generates model alignments for each required diphone segment.

7.8.1 Tracking training data

There are similarities between the process followed to track free trajectories in previous sections and the DP approach. It is, therefore, possible to retain the same two-step description and then only redefine how one accomplishes the first step. A DP algorithm now generates model alignments that segment diphone transitions into smaller parts. This procedure allows all required trajectory-based parameters (for a whole utterance) to be estimated. In a second step, we then connect all transition models, together forming a complete utterance model.

To improve the trajectory tracking of training data, we now apply a DP algorithm ("DPSegment") to perform the model alignment part. With this algorithm, we also generate transition models on a per-segment basis. Therefore, a single transition still takes place for each ASR boundary. What the DP algorithm does differently is to allow more than one "live" trajectory at a particular point. A live trajectory consists of the best model alignments generated so far, for a specific new pair of local model alignments. For each new segment and live trajectory we generate all valid alignment pairs. Each alignment pair consists of two alignment values that describe the start and end positions of a stable value. The next step is to keep only the best model alignments for each live trajectory. Finally, by comparing the cost of all live trajectories, it is possible to find

a single best trajectory for the whole utterance. In Appendix B all this functionality is described much more thoroughly. Algorithm 4 implements the main "DPSegment" functionality, using the "GetStableOptions" (Algorithms 2) and "GetPathCosts" (3) algorithms to find all local alignment pairs and cost estimates for each live trajectory.

7.8.2 Improved evaluation of test trajectories

As before, since we only estimate stable value predictors, test trajectory evaluation requires a complete segmentation of the test utterance before any assessment can take place. Oracle segmentation (see Section 7.3.3.1) provides the positions of monophone units, but model alignment values ($T1$ and $T2$) for each diphone require additional processing. In previous sections, we performed model alignment for the test utterances using the same algorithm that was used to generate the free trajectory alignments. Since the "DPSegment" algorithm searches differently for model alignments, the previous method of test trajectory alignment is not a valid approach to compare the accuracy of the new stable value predictors. Stable value predictors now have to be aligned more closely, before evaluation can take place.

Assuming sufficiently accurate references, one can achieve better stable value alignment by simply substituting reference stable values during model alignment. Both of the alignment algorithms can use reference stable values. The method in effect then allows for more degrees of freedom (per transition), when selecting the test alignments. One finds the best fit for a set of stable values rather than forcing the free trajectory alignment.

7.8.3 Results

In this section, we test whether the DP algorithm "DPSegment" can compute improved model alignments for diphone segments. Firstly, we estimate a new set of free trajectories for the training data. The DP algorithm achieves model alignment in a different manner from the previous algorithm. It is not immediately clear whether applying the "RefitStable" algorithm (as used in Section 7.7) to these models will still improve model fit. Therefore, we re-evaluate model approximation of both train and test data sets with and without stable value refitting.

Table 7.17 shows these new model approximation results (in terms of the $GWMSE_{channel}$ and $Gr_{channel}$ metrics). It is clear that just applying the DP algorithm (DP) corresponds well to previous results (Split phones). Post-processing the trajectories, now refitting the stable values at feature turning points, confirms previous trends. As before, improved

Model	$GWMSE_{channel}$	$Gr_{channel}$
Master train data		
Split phones	0.0100 (0.0382)	0.9950
DP	0.0102 (0.0394)	0.9949
Split phones + Stable refit	0.0069 (0.0288)	0.9965
DP + Stable refit	0.0066 (0.0266)	0.9967
Master test data		
Split phones	0.0098 (0.0364)	0.9951
DP	0.0100 (0.0351)	0.9950
Split phones + Stable refit	0.0068 (0.0273)	0.9966
DP + Stable refit	0.0064 (0.0237)	0.9968

TABLE 7.17: *Model fit between the free trajectories and the feature values of both the master train and test data sets using the dynamic programming algorithm and improved stable value fits. As before, incorporating the "RefitStable" algorithm provides the best model approximation (DP + Stable refit).*

tracking is possible (compare "Split phones + Stable refit" with the new "DP + Stable refit"). The "DP + Stable refit" option provides the best model approximation.

To evaluate model approximation for the predicted test trajectories, we applied the new evaluation approach. This approach (described in Section 7.8.2) found the best fit for the previous model (Split phones + Stable refit) and called the new result "Split phones + Stable refit (eval)". When we compared these two results we noted that the evaluation error became much lower with the new measurement strategy. Lastly, we added a third result: "DP + Stable refit (eval)" for the new model. In this case the DP algorithm was used for the model alignment of the training data and for the reference stable predictor value alignment on the test data set.

Model	$GWMSE_{channel}$	$Gr_{channel}$
Master test data		
Split phones + Stable refit	0.1158 (0.2217)	0.9404
Split phones + Stable refit (eval)	0.0755 (0.1618)	0.9619
DP + Stable refit (eval)	0.0632 (0.1455)	0.9686

TABLE 7.18: *Effect on accuracy of predicted test trajectories when using the new alignment strategy for the evaluation of reference stable value predictors. A considerable reduction in model error is achieved.*

Table 7.18 compares these three results. As before, this table includes values for both the $GWMSE_{channel}$ and $Gr_{channel}$ metrics. Finding the optimal model alignments (given the alignment algorithm) led to a considerable reduction in model error. The correlation (in terms of the $Gr_{channel}$ metric) instantly improved from values of 0.94 to values of about 0.96 or even close to 0.97. Estimating the stable value predictors using the DP algorithm and the new evaluation approach gave the best result. The "DP + Stable refit (eval)" experiment achieved a $Gr_{channel}$ value of 0.9686.

We noted that the DP algorithm introduced in this section was capable of improved model tracking. This result still compared well to what was achieved by the "AlignSegment" algorithm. The "AlignSegment" algorithm was much simpler to implement, a good approximation for estimating the trajectories, and did so with much lower computing resources. Because of these advantages, we mainly continued to use the "AlignSegment" algorithm in the subsequent work follow and only added a result for the DP algorithm where needed.

7.9 Free phone recognition-based segmentation

Section 6.6 discusses the evaluation of free phone recognition for ASR models trained on trajectory-based features. Similar to the work in the present chapter, oracle alignments were used as a test data segmentation for all the tests before Section 6.6 in Chapter 6. These oracle alignments were needed to estimate the trajectory models for test data features. As described in Section 7.3.3.1, this chapter details how we also used oracle alignments to segment the test data. The use of these alignments ensured that the test trajectories could be predicted for: the (1) correct phone sequence and (2) for the correct phone positions in time. As these alignments do not exist for a test utterance in practice, such alignments (and phone identities) must be inferred automatically before predicting the test trajectories. The experiments described in Section 6.6.1 yielded acceptable phone accuracies, for a correct level of smoothness. Next we tested whether free phone recognition could be used to obtain acceptable correlations for the predicted test trajectories with the new models derived, as explained in this chapter.

Model	$GWMSE_{channel}$	$Gr_{channel}$
Master train data		
Split phones + Stable refit	0.0069 (0.0288)	0.9965
No oracle	0.0069 (0.0282)	0.9966
Master test data		
Split phones + Stable refit	0.0068 (0.0273)	0.9966
No oracle	0.0068 (0.0277)	0.9965
No oracle + Aligned split	0.0066 (0.0265)	0.9967

TABLE 7.19: *Model fit between the free trajectories and the feature values of the master train data and test data sets using forced alignment to segment the train data and free phone recognition to segment the test data remains stable.*

As with the earlier sections, in order to compare the tracking of model features, Tables 7.19 and 7.20 show the re-evaluated $GWMSE_{channel}$ and $Gr_{channel}$ metrics for different trajectories of the master train data and the test data sets. To derive the trajectory models without oracle segmentation, alignments for the train data and test data sets were inferred in different ways. For simplicity, the results in this section do not include the

different model alignment strategies introduced in Section 7.8, but use free trajectory alignments to align stable values for the test data. We segmented the training data by using forced alignment with ASR models trained on the training data only. These models were trained with the latest pronunciation dictionary, where the phone labels for the stop and diphthong phone classes were split into two (Split phones). The free phone recognition system used a flat phone grammar to segment the test data. The free phone recognition used was exactly the same system as the one that generated the earlier result shown in Table 7.13, with a phone accuracy of 94.15% for the semi-tied transform. All insertion penalties were optimised on the development set.

Split phones considerably increase the complexity of a free phone recognition task. Therefore, to segment the test data, we started by using the free phone recognition system (No oracle), but optimised it by performing free phone recognition without split phones. By relying on these phone positions, we included the required phone labels (to split the phones) in the reference and force-aligned the test data with these new reference labels. This process yielded a second result (No oracle + Aligned split).

Model	$GWMSE_{channel}$	$Gr_{channel}$
Master test data		
Split phones + Stable refit	0.1158 (0.2217)	0.9404
No oracle	0.1219 (0.2315)	0.9358
No oracle + Aligned split	0.1163 (0.2313)	0.9402

TABLE 7.20: *Effect on the accuracy of predicted test trajectories when free phone recognition generates a phone sequence for test data segmentation. Forced aligning test data with a recognition-based reference (No oracle + Aligned split) restores modelling accuracy.*

It is clear from the values shown in Table 7.19 that model approximation remains stable when no oracle alignments are used. As in previous sections, this is a good indication that the new segment alignment does not introduce significant additional error for transition model evaluation. Table 7.20 shows how we continued to verify the model tracking for predicted test trajectories. The effect is clear of using free phone recognition with the larger phone set (Split phones) to find a test data segmentation. Accuracy is lower so that the $GWMSE_{channel}$ value drops lower than the baseline given in this chapter. The complexity of the phone recognition system was clearly identified as the main reason for this problem. We could find a closely similar result to the "Split phones + Stable refit" option obtained with oracle alignments. The force-alignment of the test data with a recognition-based reference sequence (No oracle + Aligned split) restored the modelling accuracy.

It can be concluded from the results presented here that it is straightforward to replace oracle alignment segmentation with free phone recognition-based segmentation for the data sets we analysed. Also, any of the results conducted in the previous sections of

this chapter would be a good approximation for an evaluation that did not use oracle segmentation.

7.10 Conclusion

This chapter describes how we optimised phone transition models to accommodate the broad range of transitions in the data better. Together with the features of choice (see Chapter 6), this work yielded the best piecewise linear representation that closely matched the feature channels of the filter bank. The goal, stated in Chapter 8, was then to apply the properties of this model to addressing the scarcity of data.

Chapter 8

Synthetic triphones

8.1 Introduction

Data augmentation approaches are aimed at increasing the amount of training data so that the parameter estimation of a speech processing system can be improved. Chapter 2 explains that, apart from the limited duration of training data, the number of training samples seen in particular contexts is a significant factor directly influencing parameter estimation. Speech systems require many parameters to model the effects of co-articulation accurately. Recently there has been renewed interest in augmenting training data by creating related synthetic versions of the actual speech data (see Section 2.5.2). Speech synthesis techniques take this strategy further and create completely new phone examples. The previous chapter confirms that feature trajectories can accurately represent speech data. This chapter discusses the aim to synthesise phone examples through the direct application of feature trajectories.

Is it possible to build phone examples with larger contexts from the trajectories of smaller (better-resourced) sub-phone units? To answer this question, we now create synthetic triphone trajectories, using a model that predicts synthetic diphone transitions (Section 8.3). As many factors influence the construction of these transitions, we require a mechanism to evaluate the quality of these transitions.

The evaluation mechanism should be closely related to the acoustic modelling of standard speech systems if one expects the analysis to inform acoustic model construction directly. We therefore create an acoustic representation of the training data and analyse how accurately such a representation matches the test data, before and after data augmentation. We expect this analysis to provide key insights into the interplay of synthetic and real phone examples. Section 8.4 presents the evaluation approach, using likelihood

evaluation. In the remainder of the chapter the approach is evaluated in the context of actual ASR system development.

8.2 Terminology

The additional terms we defined are listed in Table 8.1. These are introduced in the following sections.

Term	Interpretation
Channel trajectories	The trajectory of a single feature channel.
Parameter statistics	The estimated distribution (mean and variance) of a parameter.
Sub-phone parameters	Parameters that describe only part of a single phone transition.
Sub-phone statistics	The estimated distribution (mean and variance) of a sub-phone parameter.
Synthetic diphone predictors	Multivariate Gaussian distributions of sub-phone parameters.
Diphone synthesis	Process of creating synthetic diphone trajectories.
Example-selected HMM	An HMM used during likelihood analysis, trained on the frames of specifically selected triphone samples.
Synthetic-hybrid HMM	An example-selected HMM trained on the real and the synthetic examples of data.
Unseen triphones	Triphones that do not occur in the train data, but only in the test data set.
Rare triphones	Triphones seen only a few times in the train data, also seen in test data.
Sparse triphone set	The set of rare and unseen triphone labels that occur in development data.

TABLE 8.1: *Terminology used throughout this work.*

8.3 Synthetic triphones

The estimation of acoustic models in ASR systems requires the modelling of phones in context. In practice, ASR systems use many context sizes, from small context sizes such as biphones to larger sizes such as triphones or even quinphones – see Section 1.3. The larger context sizes in particular require large amounts of training data. Systems built with HTK [92] typically use a phone in the context of the previous and following phone (triphone) representation. In this chapter we restricted ourselves to a maximum context size of three (triphones), but the technique can be generalised for larger contexts. We extracted parameter statistics from trajectory models of the training data to construct the synthetic triphones. After selecting the correct statistical components it was possible to build synthetic diphone trajectories, triphone trajectories from the diphone trajectories and finally to convert the triphone trajectories to MFCC features. We added these MFCC features to the training data to improve acoustic model estimation. This section explains this procedure by starting with the detailed description of the parameter

statistics we modelled (Section 8.3.1). Section 8.3.2 provides more information about the statistical components we selected to create the triphone trajectories and finally, Section 8.3.3 describes all the steps necessary to create and add a synthetic triphone to the training data.

8.3.1 Sub-phone statistics

This section describes the construction of synthetic diphone predictors using sub-phone statistics. These predictors were the basis of constructing the synthetic examples in this study. As the synthetic diphone predictor term implies, we required a diphone representation of trajectories to estimate sub-phone statistics. The diphone representation was based on a set of channel transitions (one transition for every channel), which fully described a whole phone transition (as described in Chapter 5). In Chapter 7, optimising trajectory models extended this definition. Since the trajectory alignments no longer followed the ASR segmentation as strongly as in Section 7.7 in Chapter 7, extracting this set of channel transitions required more work. Section 8.3.1.1 explains the strategy to find such a diphone segmentation, where the segmentation is described in Section 8.3.1.2. This made it possible to estimate all the required sub-phone statistics from which we constructed the synthetic diphone predictors (Section 8.3.1.3). The synthetic diphone predictors made it possible to synthesise the diphones.

8.3.1.1 Channel trajectory segmentation

The new diphone representation required additional alignments for all the channel transitions of a segment. In essence it was necessary to start and end alignments for each channel transition. Accordingly, we re-segmented the trajectories of each feature channel to find the channel transitions of each trajectory. Figure 8.1 depicts this segmentation process for two feature channels. If there is a sequence of phones /a/,/b/,/c/,/d/ and phone boundaries (black vertical lines) between the frames of each phone, then three diphone transitions (/a=b/,/b=c/,/c=d/) can be drawn. Previously, each diphone transition consisted of the 50% of monophone frames closest to the phone boundary (Section 5.3.1). Similar to the representation shown in Figure 5.2, again we found the $T1$ and $T2$ alignment values for each feature channel i. Figure 8.1 shows how the feature trajectories are segmented for one diphone transition (/b=c/), introducing ch_{start} and ch_{end} parameter values for each channel i. Each segmentation occurs at exactly the centre of a stable value. When considering the second transition alignment value for channel i as $T2(i, /a = b/)$ and the first transition alignment value for channel i as $T1(i, /b = c/)$, then:

$$
\begin{aligned}
ch_{start}(i, /b = c/) \quad & \text{Centre between } T2(i, /a = b/) \text{ and } T1(i, /b = c/) \\
ch_{end}(i, /b = c/) \quad & \text{Centre between } T2(i, /b = c/) \text{ and } T1(i, /c = d/)
\end{aligned} \tag{8.1}
$$

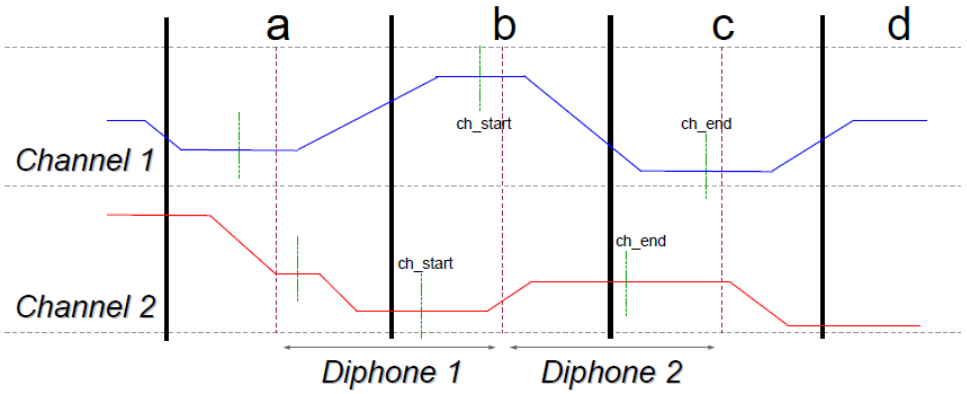

FIGURE 8.1: *Channel-specific re-segmentation of trajectories into diphones (green vertical lines represent new ch_{start} and ch_{end} alignment values).*

Figure 8.1 indicates each of the new ch_{start} and ch_{end} alignment values marked with green vertical lines. Lastly the midpoint ($ch_{mid}(i, /b = c/)$) of each aligned channel i for the diphone transition label /b=c/ was calculated as:

$$
ch_{mid}(i, /b = c/) \quad = \quad \frac{ch_{start}(i, /b = c/) + ch_{end}(i, /b = c/)}{2} \tag{8.2}
$$

The next section explains how we use these new alignment parameters to find a diphone segmentation.

8.3.1.2 Diphone segmentation

The previous diphone segmentation strategy to find the 50% of monophone frames closest to the phone boundary did not allow proper segmentation across all channels for the trajectories in Section 7.7. During the optimisation process, we updated the time alignments for these trajectories. The two-channel example in Figure 8.1 makes this clear. For channel 2 the diphone transition /b=c/ starts before the diphone segmentation (dotted vertical line) and if this transition is included as it is to estimate parameter statistics, it would produce invalid estimates. To circumvent this diphone segmentation problem, we prefer to define separate diphone segments of longer durations so that all the channel transitions could take place within the segmentation boundaries. To describe

the new diphone segmentation, we define the following parameters:

$$
\begin{aligned}
diph_{start} &\quad \text{Lowest } ch_{start} \text{ value across all channels } i \\
diph_{end} &\quad \text{Greatest } ch_{end} \text{ value across all channels } i \\
diph_{mid} &\quad \text{Mean across all } ch_{mid} \text{ values: } \mu(ch_{mid}(i, /b = c/)) \\
dur1 = diph_{mid} - diph_{start} &\quad \text{First half-duration of diphone alignment} \\
dur2 = diph_{end} - diph_{mid} &\quad \text{Second half-duration of diphone alignment}
\end{aligned} \tag{8.3}
$$

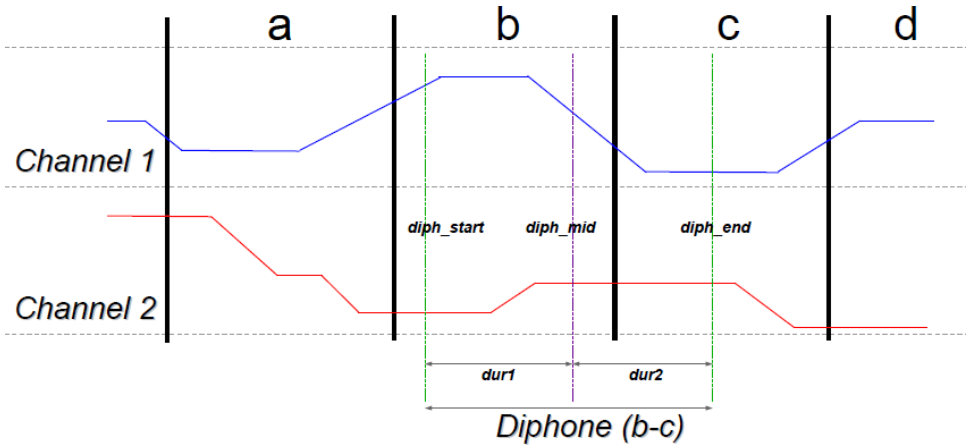

FIGURE 8.2: *New diphone segmentation (boundaries $diph_{start}$ and $diph_{end}$) for the diphone transition /b=c/ considering all channels i.*

Figure 8.2 shows the new diphone segmentation of the diphone /b=c/. The diphone boundaries ($diph_{start}$ and $diph_{end}$) are represented by vertical green lines. At the centre of these boundaries, the $diph_{mid}$ alignment separates the $dur1$ and $dur2$ values, together forming the total duration of the new diphone segment. The new diphone segmentation then yields all the parameters one needs to model complete diphone trajectories.

8.3.1.3 Synthetic diphone prediction

This section describes the synthesis of diphones using sub-phone statistics. To this end, first we presented all the sub-phone parameters required and modelled the statistics. After creating multivariate predictors for the sub-phone statistics of individual feature channels, we could construct the synthetic diphones. The set of required sub-phone parameters from separate channels was as follows:

$$
\begin{array}{rl}
S1(i) & \text{Feature value at initial stable value of channel i} \\
S2(i) & \text{Feature value at final stable value of channel i} \\
R1(i) = \frac{T1 - diph_{start}}{dur1} & \text{The relative start position of the transition for channel } i \\
R2(i) = \frac{T2 - diph_{mid}}{dur2} & \text{The relative end position of the transition for channel } i \\
dur1 & \text{First half-duration of diphone alignment} \\
dur2 & \text{Second half-duration of diphone alignment}
\end{array} \tag{8.4}
$$

We employed two multivariate Gaussians with 53 dimensions each to model the sub-phone parameters across all diphone examples /b=c/ seen in the training data. These sub-phone parameters may be estimated from different samples when constructing rare or unseen phones. We grouped all the sub-phone start parameters $S1$, $R1$ and $dur1$ to create the 53 channels of the first Gaussian and similarly assigned all the sub-phone end parameters ($S2$, $R2$ and $dur2$) to 53 channels for the estimation of the second Gaussian. Once these had been estimated, synthetic examples for diphone /b=c/ could be found.

The first step in the diphone synthesis approach was to create synthetic trajectories for the required diphone label. We sampled each of the two selected multivariate Gaussian distributions to generate an example for the diphone /b=c/. This process obtained sample values for each sub-phone parameter. The sampling process had to comply with the restrictions pertaining to valid alignment sample values. We list these restrictions at the end of this section. Figure 8.3 depicts a completely new synthetic diphone segment. This new diphone mimics the diphone in Figure 8.2, but as intended does not duplicate the initial /a=b/ transition of channel 1, since we are currently creating diphones in isolation. (Note that when creating these diphones in isolation, the seg_{start} parameter is set to a value of 0.) After finding the following parameter values, the trajectory of the new synthetic diphone could be described as:

$$
\begin{array}{rl}
seg_{dur} = dur1 + dur2 & \text{Duration of the synthetic trajectory} \\
T1(i) = R1(i) * dur1 & \text{Frame index at the start of the transitions for channel } i \\
T2(i) = dur1 + R2(i) * dur2 & \text{Frame index at the end of the transitions for channel } i \\
S1(i) & \text{Sample value of first stable value for channel } i \\
S2(i) & \text{Sample value of second stable value for channel } i
\end{array} \tag{8.5}
$$

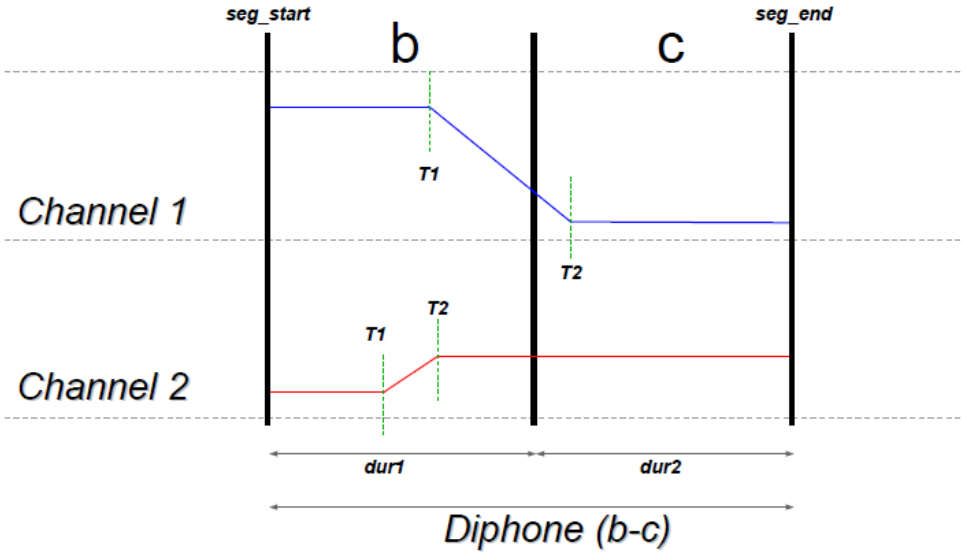

FIGURE 8.3: *Synthetic diphone trajectories for the synthesised diphone /b=c/.*

Only the model alignment parameters where $T1 < T2$ occur were used for synthetic diphone examples. Sample values for the multivariate Gaussians might contain examples where $T1 > T2$. For this reason, we restricted ourselves to using only the sets of sample values where $T1(i) < T2(i)$ for all channels i. Similarly, the start of transition ($T1$) or the end of transition ($T2$) values might fall beyond the diphone boundaries (seg_{start} and seg_{end}) in Figure 8.3. As a further restriction, therefore, we ensured that $T1(i) > seg_{start}$ and $T2(i) < seg_{end}$ for all channels i.

8.3.2 Component selection to create new n-phone

A complete utterance consist of many transitions. Transitions can be modelled in different ways and many model parameters can be defined for each of these transitions. In this work, we chose a limited number of parameters, which provided a compact description for diphones. The transition model definitions (Sections 5.3.1 and 8.3.1) specified these selected model parameters. Each parameter described a sub-component of a trajectory, given the examples of training data seen in the same context. It follows that one could estimate the reference parameter values of any context size seen in the training data. For example, Chapter 5, Section 5.5.1.1, explains how we employed reference stable values (with a biphone context size) to test trajectory estimation for stable values. Different context sizes could be evaluated. We continued to use biphone context sizes for all sub-phone parameters in the current chapter. As in Section 5.5.1.1, we estimate model parameters using the chosen set of references (from Section 8.3.1.3). This time, we use a full covariance matrix, as we expected this to result in more powerful predictors. Given the description of the modelling choices above, it is clear that there are many additional

parameter options for generating synthetic examples for the phones of an utterance. In this work, we chose one such option for which we constructed the multivariate predictors.

The test ASR systems we built in the experiment of this chapter used cross-word triphone models. Acoustically, these systems trained an HMM model for the feature values of every phone example seen. By specifying left and right contexts for monophone labels, the examples of a particular monophone were divided into many more (triphone) classes. We constructed synthetic examples of specific phones so as to improve the estimation of ASR models. We had various options at our disposal to realise any single phone example by means of trajectory models. For example, the first step in building a triphone (phone with a left and right context) was to find the trajectories of two diphone transitions and then combine and sample them. This process generated the required feature values.

Specifically, as regards the monophone labels /a/, /b/ and /c/, we could construct one triphone example /a-b+c/ by concatenating two diphones. This process required two examples of the monophone /b/, one in the right context of /a/ (/a=b/) and another in the left context of /c/ (/b=c/). We used complete transition models (diphones) in the experiments and combined them to form a triphone. This made it possible to combine examples of parameters with different context sizes to form diphones. The subsequent sections describe how we constructed the synthetic triphone examples using two diphone transitions.

8.3.3 Generating phone examples

This section describes the complete process of creating the synthetic features and augmenting the ASR training data. We predicted phone examples (where examples of a particular context were limited) and generated features for synthetic versions of these phones. We created synthetic features and added them directly to the standard set of ASR training features. The detailed process of generating the synthetic training data can be summarised in the following steps:

1. Estimate free trajectories from training data: one needs all the associated monophone examples that represent specific diphones to generate new synthetic examples for a required set of triphones. In the training data, these phone examples occur within complete utterances. As explained in Section 7.3.3.2, we construct complete utterance models to create the trajectories of each utterance.

2. Select the n-phone components: find the two diphone labels required to create a specific triphone. Section 8.3.2 introduces this procedure; for example, to generate triphone model /a-b+c/ one selects diphone transitions /a=b/ and /b=c/.

3. Context fallback: use the monophone label if there are insufficient examples of a diphone in the training data. For example, if diphone /a=b/ occurs fewer than n times, rather use monophone label /b/ to represent all the diphones of /b/ in any left context /*/ (/*=b/).

4. Find the channel trajectory segmentation: find the channel transition alignments for the trajectory of each selected diphone label (see Section 8.3.1.1).

5. Perform diphone segmentation: obtain the parameters that describe the diphone representation to be used for estimating the sub-phone statistics, given all available training examples.

6. Synthesise the diphones: create synthetic diphone trajectories, following the procedure in Section 8.3.1.3. Start by extracting sub-phone parameters from each diphone example in the training data. Secondly, model the sub-phone statistics by using multivariate Gaussian distributions, thus creating synthetic diphone predictors. Thirdly, sample the synthetic diphone predictors (applying the required restrictions) and finally find the parameter values that describe the new synthetic diphone trajectory.

7. Synthesise the triphones: connect the synthetic diphone trajectories (see Figure 8.3) for two diphones to form triphone features. As before, this is simply a matter of following the procedure in Section 5.4.1 to connect the channel transitions for each channel.

8. Frame selection: given the n-phone (triphone in the present case) trajectories, select only the frames comprising the specific phone example that is now successfully modelled in context. Using the diphone segmentation from which we created the triphone trajectories, these frames are the last 50% of the frames from the first diphone concatenated to the first 50% of the frames from the second diphone trajectory.

9. Synthesise MFCCs: sample the selected frame indices of the previous step and convert these features to MFCCs.

8.4 Likelihood evaluation

Synthetic training examples increase the amount of training data for training the ASR models. Although large numbers of synthetic triphones can be generated by using the technique described in Section 8.3, not every synthetic phone example will necessarily improve the system's accuracy. As mentioned in Section 2.5.2, the quality of synthetic

examples should first be determined. If the quality is not optimal, the quantity has to be controlled. Errors tend to accumulate when adding sub-optimal phone examples to a speech system and may lead to reduced performance. The selection of "correct" synthetic examples to be added is therefore crucial to ensure the estimation of improved acoustic models. Section 8.4.1 contains the definitions of example-selected HMMs to represent the new (augmented) set of training data for a particular triphone label. After training the same type of HMMs for the baseline (using only original training examples), one could next evaluate how well each model matched the frames of a test phone example. Section 8.4.2 describes the use of HTK to calculate such likelihoods. We measured the difference (shift) in likelihood of the test sample when comparing the old and new models (Section 8.4.3). Finally, choosing the example-selected HMMs to provide the best match and generalising across different test evaluation examples indicated which synthetic examples improved model estimation.

8.4.1 Example-selected HMMs

HMM-based acoustic model training usually takes frame-based features and a sequence of phone labels as input into the training process. Since we were interested in controlling precisely which phone examples a particular HMM model was trained on, it was best to avoid working with complete utterances. Instead we pre-processed the training data and chunked all the utterances into single phone units. In this way, the precise phone locations were known. Such a chunking procedure required a set of known phone alignments. We trained example-selected HMMs directly on the selected frames of single phone examples.

For results to be comparable, example-selected HMMs should closely follow the design of standard HMMs employed in normal recognition recipes. We used a model topology of three left-to-right emitting states. The state distributions were GMMs with a limited number of mixture components. The example-selected HMMs first trained triphone models where incremental mixture increments took place. We built example-selected HMMs of two types: (1) an untied-triphone and (2) a tied-triphone. By evaluating the untied-triphone models, we intended firstly to determine whether specific triphones yielded a better model estimation. In a tied-triphone model, the related triphone labels are also part of the training examples for that particular model. The tied-triphone models were also evaluated because we wished to determine whether optimising the data sufficiency (using decision tree clustering) would still provide gains with the addition of synthetic phone examples.

8.4.2 Log likelihoods for test phones

We required an algorithm to estimate HMM likelihood, given the test features, to evaluate the example-selected HMMs. These test features describe phone examples taken from a test data set. HMMs can match the same features in many ways. The state-transition probabilities allow an HMM to have many optional matches for a specific sequence of frames. This section focuses on finding the highest likelihood (best match). The HVite decoder tool in HTK [92] has the capability to calculate this maximum log likelihood score for each example-selected HMM we wanted to evaluate.

In HTK, the HVite decoder finds the best path through a network of possible paths for the HMM state sequences. A *forced alignment* constrains the number of possible paths to include a sequence of labels that only matches the transcription of a particular utterance. One generally requires word-level transcriptions and a pronunciation dictionary to use the decoder in *forced alignment* mode. This dictionary and transcription set could then generate all the possible phone strings for the pronunciation of each utterance. HVite finds each possible recognition path by performing an HMM lookup for each phone in the strings of phone labels and links all the relevant HMMs together as nodes. HMM nodes consist of the emitting model states connected by the model topology arcs. The decoder finds the best path through this graph of all possible states by using a *token passing* algorithm [92]. Each token represents a partial evaluation path and keeps track of the sum of log likelihoods (cost) as the algorithm traverses a particular path. The algorithm ends by choosing the path (state sequence) that provides the best (maximum) log likelihood.

For any state j of a given model M and an observation o it is possible to calculate the log likelihood of all frames of o assigned to that specific state during HMM alignment of the full observation. We define a state-specific log likelihood $\phi(j)$ which is the sum over all the relevant frames.

This likelihood is recursively calculated as in [92]. While being in state j at time t, the maximum log likelihood $\phi_j(t)$ of observing speech frames o_1 to o_t is calculated using:

$$\phi_j(t) \quad = \quad \max_i\{\phi_i(t-1) + log(a_{ij})\} + log(b_j(o_t)) \tag{8.6}$$

where a_{ij} is the state transition probability from a previous state i to the state j, $b_j(o_t)$ is the probability of observing o_t while being in state j and $\phi_i(t-1)$ represents the maximum log likelihood value of being in state i after observing the first $t-1$ frames.

Then

$$\phi_M(j) = \sum_k \phi_j(o_k) \tag{8.7}$$

where o_k represents all frames assigned to state j of model M during HMM alignment.

In the experiments to follow, we wanted to evaluate the log likelihood of the sequence of frames describing a single phone example. To this end, we constrained the HTK dictionary lookup to generate only a single phone label and provide only the matching frames as a single utterance. In this way, the above search was simplified so that the decoder only considered all the possible state sequences that could be formed from the emitting states of the selected model for the observed frames.

8.4.3 Evaluating the log likelihood shift

As described in the previous section, we could find a single log likelihood value to describe the match for each emitting state of a particular HMM and the frames of any given test phone example ($\phi_M(j)$ for model M in state j). The HVite decoder assigned specific frames to match every state (j) in the chosen state sequence. In essence, we were interested in finding the difference that adding a single synthetic phone example made to the estimation of the matching HMM label. To measure these differences, we first trained a baseline HMM model, using only the phone examples originating from the real train data. In addition, we kept track of the exact number of phone examples (of the same label) used to estimate this particular HMM. A second synthetic-hybrid HMM could be trained next incorporating all of the real training phone examples for a particular phone label and also an additional number of synthetic phone examples. We recorded the total number of phone examples (real and synthetic) on which we estimated the new model.

Next we could calculate the shift in log likelihood (γ) for each emitting state j, when observing the speech frames of a test phone example, first calculating $\phi(j)$ for each HMM state:

$$\gamma = \phi_{hybrid}(j) - \phi_{baseline}(j) \tag{8.8}$$

where $\phi_j(baseline)$ is the state-specific maximum log likelihood of the test phone example when evaluating the baseline HMM. Similarly, $\phi_{hybrid}(j)$ is the state-specific maximum log likelihood of the test phone example when evaluating the synthetic-hybrid HMM.

Equation 8.8 defines a positive γ when the synthetic-hybrid model yields an improved, state-specific, maximum log likelihood (better than the baseline).

8.4.4 Choosing the best predictors

Up to this point, the shifts in the log likelihoods (γ values) we measured between baseline and synthetic-hybrid HMMs were for separate emitting states of single phone examples. From here onwards, grouping the γ values across HMM states and different phone examples made it possible to obtain generalised results. Specifically all the emitting states of test phone examples with the same label were clustered together. One could make many comparisons for different types of HMM-based acoustic models. In this work we specifically chose the two options (models for untied-triphones and tied-triphones) mentioned in Section 8.4.1. When analysing untied-triphone models, the phone examples were clustered for each separate triphone label. For the tied-triphone analysis, we generalised further by clustering all the phone examples mapping to tied-state clusters as formed by the tied-triphone models. Estimating a mean value to represent the shift of each group then enabled further comparisons.

It is straightforward to define the mean γ values for untied-triphone models. For the same triphone label, we grouped the γ values for every test triphone example and emitting state. The tied-triphone model required an additional step to establish these group statistics. Similar to the tied-triphone HMMs in standard ASR recipes, each of the example-selected HMMs could be analysed to discover the tied-state clusters of triphone labels. In the models used in this study there are three sets of tied clusters, one set for each emitting state number. For each of these sets, we tied states among different triphone labels. These state clusters differed between the baseline and synthetic-hybrid HMMs. Thus the effect of the additional synthetic data was that the model training algorithm had to find another set of tied-state clusters. We always grouped states (and derived the statistics) to represent a specific model comparison by using the tied-state triphone labels of the synthetic-hybrid HMM instead of the baseline HMM. In doing so, additional error introduced into the clustering process of the synthetic-hybrid HMMs would be implicitly modelled within the mean γ values. (To clarify: a test sample will be evaluated against the correct tied-state model per system; it is only during mean calculation that the tied states from one of the two systems are used to combine results.)

Depending upon the particular analysis, it is possible that no triphone examples in the test data match the label of the synthetic-hybrid HMM that needs to be evaluated. Given the clustering process, it is possible to examine the effect of including synthetic examples for such unseen triphone labels. We could use the mean γ values for all state examples of

the γ value group that an unseen triphone label mapped to as an indication of synthetic example quality. The tied-triphone HMMs by definition include examples of similar, but different triphone labels in the same group. These triphone labels may also include synthetic examples of other triphones in that same group. Therefore the tied-triphone HMM strategy might provide a better indication of overall system performance (closer to the typical context-dependent modelling of standard ASR systems). On the negative side the complexity of pooling various real and synthetic phone examples (with different triphone labels) could degrade model estimation. Section 8.8 discusses the implications of this interplay, which increases the complexity of synthetic example selection.

8.5 Experimental setup

This chapter describes the experiments we conducted to augment data by adding selected synthetic phone examples to improve the representation of a limited training data set. The purpose of this section is to introduce all the implementation choices required to implement this strategy. The selection of the data sets is described in Section 8.5.1. We require trajectory models (Section 8.5.2) to construct the synthetic examples and we use these synthetic phones in various ways to augment the training data (Section 8.5.3).

8.5.1 Data sets and segmentation

We specifically intend to work with a limited subset of training data for the data augmentation experiments. A repartitioning of the well-resourced data sets described in Chapters 6 and 7.1 was required. These data sets contain a total of 4 974 utterances with an approximate duration of 3 hours and 20 minutes. Since utterances are only one to three words long, this translates into roughly 25 utterances per minute. Table 8.2 shows the detailed segmentation of the well-resourced data set we used previously.

Data set	**#Utts**	**#Dur (min)**
Master train	4072	163
Master test	902	36
Dev: train	3199	128
Dev: test	873	35

TABLE 8.2: *The number of utterances in our previous well-resourced data sets, as well as their estimated duration in minutes.*

In the current work, the aim was to apply the technique that had been developed to supplement ASR data sets of limited size. This is a different strategy from previous analyses, where it was more important to ensure that enough examples of all the diphones we wanted to analyse were present. A random selection of utterances describes the

outcome of the typical data collection effort better. To construct the limited data sets used in these study, we partitioned the available utterances randomly into non-overlapping data sets of 80%, 10% and 10% to form the new train, development and test sets, respectively. Each set's size was determined on the basis of duration (instead of the number of utterances). Furthermore, for simplicity every training data set had to include at least one example of each monophone occurring in the test data sets. Utterances were only allowed to be part of the development or test data sets if all the monophone labels were seen more than $n = 20$ times in all the available data.

In practice, we partitioned the training data into a required number of non-overlapping subsets. Given a predetermined duration of a subset, we randomly selected the utterances of each subset until the subset reached its predetermined duration. Lastly, we checked that at least one example of every monophone occurring in the test data was present in each subset. Next we divided the training data into five parts, applying a duration cut-off of 31.17 minutes. We chose to use one of the five parts in the experiments that follow and to denote this data set as $train_{30}$.

8.5.2 Feature trajectories

Step 1 in the procedure to generate synthetic phone examples (see Section 8.3.3) required a complete set of trajectory models for the training data. To obtain these trajectories for the newly selected $train_{30}$ data set, we re-used exactly the same utterance models mentioned in Chapter 7. The configuration during the estimation of these utterance models ensured that we sampled each channel of 26 filter bank channels at a 5 ms frame rate. For every utterance, these raw filter bank outputs were converted to log filter bank values and the mean was subtracted from each feature channel. ARMA filtering with a window size of six frames was applied and finally piecewise linear trajectory models with three pieces per diphone were estimated to model all the transitions of a complete utterance. Further trajectory optimisation was applied: as in Sections 7.5 and 7.7, improved phone transitions were obtained by splitting phones and optimising stable values. Specifically, we re-used the final trajectories mentioned in Section 7.7 to evaluate all the likelihoods for synthetic example selection.

Section 7.3.3.1 explains that "oracle" alignments are part of the estimation process for the piecewise linear models from which the best trajectory model was selected. Section 7.9 explains that later, we could replace oracle alignments with blind recognition. In the work described in this chapter, we decided to follow a similar strategy. Since utterance models with an oracle segmentation already existed for all the utterances listed in Table

8.2, we could re-use these trajectories for any subset of utterances without requiring any re-segmentation.

8.5.3 Synthetic phones

Resource limitations resulted in poor modelling of some contexts seen in the training data. Consequently, we expected that the new synthetic examples would improve the situation. Even though the quality of synthetic phone examples might be sub-optimal, in rarely seen or unseen contexts these additional examples might alleviate the parameter estimation problem when training acoustic models. This section describes the approach to targeting the poorly modelled contexts of the specific train and test data sets in this study. Section 8.5.3.1 gives detail of the criteria for label selection, Section 8.5.4 describes the practical choices we made for implementation and Section 8.5.5 explains how we controlled the quantity of the synthetic data.

8.5.3.1 Label selection and overlap

In this section, we only analyse triphone examples for contexts seen in the test data, and only target labels that are not well represented in the training data. In order to understand the resulting triphone distribution better, we started by grouping the triphone labels into two categories. Given the list of triphones in the test data and for a particular train data set, we determined which triphones:

1. did not occur in the train data set (unseen triphones)
2. were seen only a few times (rare triphones).

The goal of this step was to find the list of triphone labels for which synthetic examples should be generated, as well as the number of examples required for each and every label. Since a random selection of utterances formed the test data set (see Section 8.5.1), we expected that one would see similar phone distributions in this data set compared to the phone distribution of the training data. Table 8.3 and Figure 8.4 provide more information, specifically for the unique triphone labels (Label type) of the $train_{30}$ data set in relation to the development (Dev) and test data sets. The unseen and rare triphone categories provided counts for the number of triphones of these types seen in the test data. The overlap of labels within the development data is also indicated (Unseen observed in dev & test and Rare observed in dev & test) in these two categories. Lastly, we compared the counts of each category as a percentage (Percentage) of the number of

triphones from a larger category. We provided the percentages for the data sets in terms of all the data and the percentages of the rare and unseen category labels in terms of the test data only.

Label type	**#Triphone Labels**	**Percentage**
All	4714	100
$train_{30}$	3538	75
Dev	2896	61
Test	2833	60
Unseen observed in test	651	23
Rare observed in test	1999	71
Unseen observed in dev & test	190	7
Rare observed in dev & test	1554	55

TABLE 8.3: *The number of triphone labels per label category.*

It is clear from Table 8.3 that only 75% triphone labels are part of the training data. Given the development and test data sets, 25% of labels are not seen in the training data at all. Furthermore, in the development and test data sets about 60% of the triphone labels occur. Six hundred and fifty-one test labels were unseen in the $train_{30}$ data and at a threshold of 15, 1 999 labels formed part of the rare category. This large number of rare category triphones (71% of the test labels) is best explained in Figure 8.4. Only 161 triphones occur more than 15 times in the $train_{30}$ data.

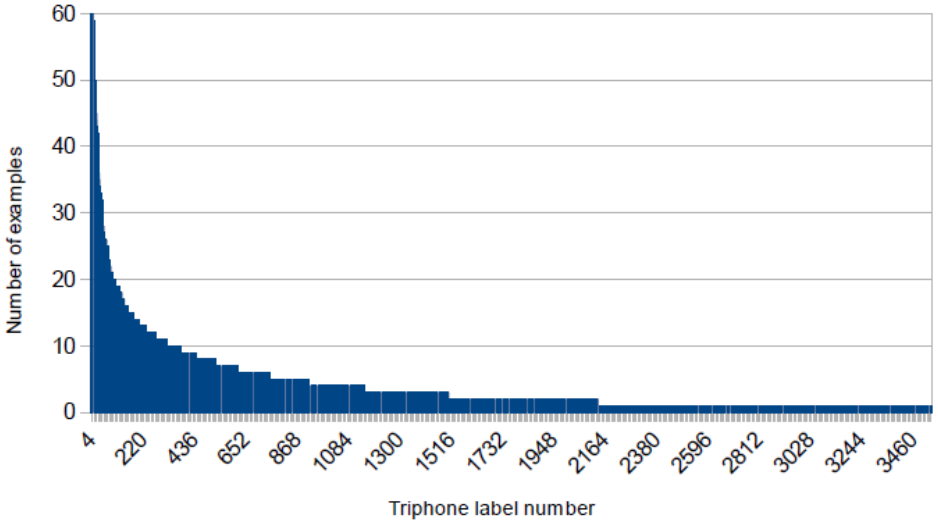

FIGURE 8.4: *Number of examples for triphone labels in $train_{30}$, showing that only 161 triphones occur more than 15 times.*

As we intended to use the development data set for analysing the synthetic examples, the overlap of the rare and unseen label categories and the labels seen in the development data was also of interest. Only 190 labels of the unseen category labels occurred in the development data, but there was a much larger percentage of the rare label category (78%). We named this specific set of labels that are also seen in the development data

the sparse triphone set. From this point on we focused on generating examples for specific triphone labels occurring in the sparse triphone set only, one example at a time.

8.5.4 Constrained synthetic examples

The procedure in Section 8.3.3 describes the steps we followed to create a completely new MFCC utterance. Such an utterance contains a single synthetic triphone. In practice, there are a few practical considerations when implementing this process. This section lists these considerations and discusses the implementation choices we made for the experiments discussed in this chapter. We describe each of the following items:

- Context fallback threshold: the choice of threshold for diphone context fall-back (see (3) in Section 8.3.3).
- Predictor distributions: the use of full co-variate Gaussians as synthetic diphone predictors.
- Sampling strategy: the sampling strategy to generate samples given the synthetic diphone predictors.
- Split phone triphone synthesis: the synthesis of triphones for split phone trajectories (introduced previously in Section 7.5).

8.5.4.1 Diphone fallback

The third step of the process of generating synthetic data required at least that n diphone examples be present before the algorithm modelled these examples in a diphone context in sub-phone statistics. In the experiments described in this chapter, we set this threshold at a value of $n = 3$. One of the things that a threshold of $n > 1$ ensures is that sub-phone parameters do not have zero variance. A sub-phone parameter with zero variance may result in synthetic diphone predictors where the imposed constraints on samples to synthesise diphones cannot be met.

8.5.4.2 Covariance modelling

Section 8.3.2 states that using a full covariance matrix may improve the ability of synthetic diphone predictors to predict more accurate samples. The covariance between sub-phone parameters encodes additional information about specific phone transitions. In this work, we chose to work with a full covariance matrix for each synthetic diphone

predictor where the initial sampling process was successful in generating a set of valid samples. There are constraints on the allowed time alignments that generate valid samples. For some examples it was necessary to employ a more elaborate sampling strategy to ensure that each diphone was synthesised. While more elegant solutions are possible, we implemented a brute force search for valid samples in order to evaluate the larger concept.

8.5.4.3 Re-sampling

The brute force sampling process works as described below. If the algorithm could not meet the imposed constraints on the samples used for constructing a complete diphone, the first step was to try to re-sample. We (randomly) re-sampled each of the two synthetic diphone predictors up to a maximum of 50 times. Re-sampling may, however, occur more than 50 times until both of the sub-phone parameters $dur1$ and $dur2$ are positive. We kept track of the samples where the constraints for the lowest number of channels failed. The re-sampling step might exceed the re-sampling threshold, and the final samples would remain invalid. To rectify this situation, we selected the samples (from the kept inventory) where $dur1$ and $dur2$ were positive, and the smallest number of channels remained in error. For these channels only, we corrected the invalid parameter values by breaking the covariance constraint. Next, we generated samples for one parameter at a time, using only the mean and variance to describe the sample distribution. Updating each invalid parameter value with the first valid sample finally yielded a valid sample for all channels of the synthetic diphone predictor.

8.5.4.4 Split phones

The previous chapter states that split phone models result in improved trajectories. Reusing trajectories of this type as features while training ASR models required a step of conversion. We recombined any split phone triphone labels to form the original triphone labels. To accomplish this, we extended the functional descriptions (steps 2, 6, 7 and 8 of the procedure mentioned in Section 8.3.3) to work for a sequence of (instead of only two) diphones. Below is a list of the additions:

- In step 2 (n-phone components), we use three or more diphone labels to form a single triphone model. For example, to generate the triphone model /a-b+c/ now one may have to select /a=b1/, /b1=b2/ and /b2=c/ if the phone /b/ was split into two parts (/b1/ and /b2/).

- Since there are at least three diphones, synthesise three or more diphone trajectories and no longer just for two diphones (step 6).
- By connecting the diphone trajectories of all the diphones in sequence, synthesise a single triphone (step 7).
- For frame selection (step 8), use the last 50% of the frames from the first diphone, all the frames of the centre diphones and only the first 50% of the frames from the last diphone.

The above configuration enabled us to generate all the synthetic triphone examples for the work described in this chapter. Synthetic example selection must occur before improved models can be trained. The selection process is described in the next section.

8.5.5 Adding synthetic examples for HMM estimation

The strategy to select synthetic examples was based on a log likelihood evaluation of development data examples. Section 8.4.4 explains how a mean γ value we estimated for state-specific log likelihoods enabled us to choose which acoustic models represented the development data better. The goal was to find which synthetic-hybrid HMMs outperformed the baseline HMMs for the same phone classes. To this end, we evaluated example-selected HMMs of untied-triphones as well as of tied-triphones. Section 8.5.3.1 already described which triphone labels we analysed. The experiments described next ensure that all the labels from the sparse triphone set occurred at least a minimum number of times in the training data set. The values of this threshold varied between 1 and 15. Synthetic examples were added to the training data by drawing from a pool of synthetic triphones generated earlier. As stated in Section 8.4, controlling the quantity of the training data enabled us to choose the synthetic examples that were expected to improve the acoustic modelling.

The experiments described in Section 8.6 required us to train many synthetic-hybrid HMMs, which was done in batches. We performed our analysis on models trained for each label in the sparse triphone set. The first two batches of models we trained were a set of baseline models for the untied and the tied-triphone representations. For the tied-triphone model set, the goal was to train models for these same sparse triphone label categories. State-tying does, however, require examples from other triphone labels in the training data. We ensured that all the tied-state models required trained successfully. In the experiments, we evaluated models with different numbers of mixture components per state. Each particular number of mixture components required another batch of

example-selected HMMs. We worked with models containing anything from one mixture component per state up to eight mixtures per HMM state.

Additional experiments were conducted to analyse the effect of including different numbers of synthetic phone examples for model estimation. In this work, the strategy we followed was to start training all model sets for the unseen triphone label category. We trained a batch of models by adding only one synthetic example for every unseen label; in the next batch, two synthetic examples were added, and so on. In total, we constructed 15 batches of these synthetic-hybrid models. The model sets for the rare label category worked similarly; except that we started with an example count of two. In that batch of models, we added one synthetic example for each rare label category phone in the training data where only one real example was present. This step brought the total example count of the labels of the rare label category to at least two examples in the training data. As with the unseen labels, we also constructed a set of models for each minimum number of rare label counts, up to a count of 15. Lastly, but only for the tied-triphone HMMs, we also had to construct HMM sets where we combined the unseen and rare category labels. This was because examples of both categories could be added to estimate the same tied states. Similar to the model sets for the rare category labels, we started at a label count of two, and trained the model sets for the combined category up a required example count of 15.

It was now possible to analyse the log likelihoods of the test data, given these acoustic representations.

8.6 Analysing likelihoods

We divided the analysis of example-selected HMMs into two parts. Firstly, we evaluated these models, using the development data (Section 8.6.1). The aim of these experiments was to discover whether any synthetic-hybrid HMMs showed improved representation for the speech examples of the development data set. We tested the sensitivity of the analysis by using different numbers of mixture components and considered the implications of example sufficiency and data sharing, using tied-triphone models. This revealed the number of synthetic examples that would improve the model estimation for development data. Finally, Section 8.6.2 investigates whether the testing of synthetic example selection could be generalised (using development data) so that the likelihood of unseen test data would improve accordingly.

8.6.1 Development data

Section 8.5.5 explains that we started by training all the different example-selected HMMs for untied-triphones. In addition, we analysed the triphone labels of the sparse triphone set only. The plots below depict the mean difference in log likelihood (mean across all the state-specific γ values) of each of the 1 744 sparse triphone labels, when comparing the baseline and synthetic-hybrid HMMs. Furthermore, we ranked these mean difference values for each experiment from high to low to compare the overall effect of the synthetic example selection process across all the labels. Values above zero show that, compared to the likelihood of the baseline model, the best-performing synthetic-hybrid model option for this triphone was a better match to the development data. Figure 8.5 shows the results of such an analysis using various example-selected HMM sets and mixtures. A mean difference in log likelihood of zero occurs where no synthetic-hybrid HMM performs better than the baseline and therefore the baseline model is used to generate the best likelihood.

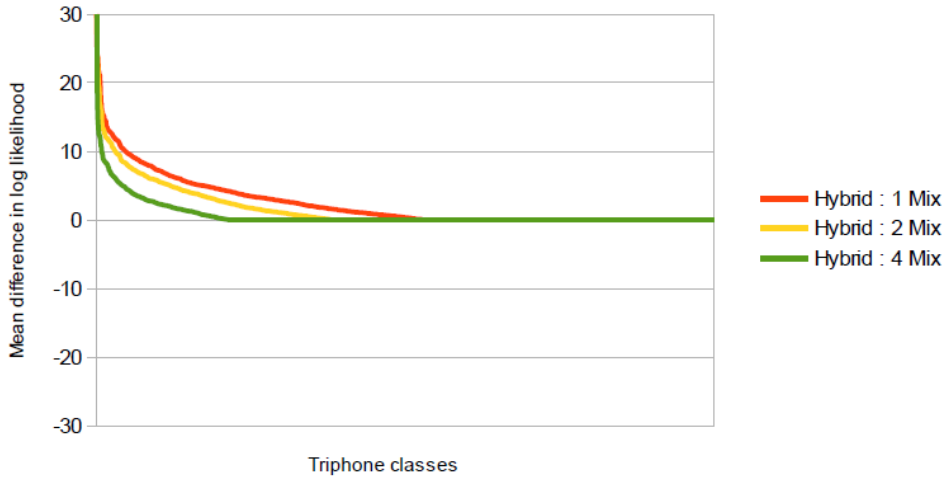

FIGURE 8.5: *Comparing the improvement in the mean log likelihood (γ values) of untied HMMs for different numbers of mixtures. Fewer triphone classes result in improved triphone modelling for "Hybrid : 2 Mix" and "Hybrid : 4 Mix" models.*

Before we interpreted these results in more detail, one additional experiment shown in Figure 8.6 made it possible to understand the size of the likelihood increase better. This experiment compared the mean difference in log likelihood between four-mixture baseline and one-mixture baseline models. The result of Figure 8.6 clearly shows that not all triphones analysed train better four-mixture models than one-mixture models. A number of labels (14%) train degraded baseline four-mixture HMMs. For the remaining 86% of sparse triphones, the mean difference in log likelihood has a comparible size to the differences detected for the experiments depicted in Figure 8.5.

FIGURE 8.6: *Comparing the difference in the mean log likelihood between the untied HMMs of baseline single mixture and baseline four mixtures. Not all triphones train better four-mixture models than one-mixture models.*

All three of the experiments in Figure 8.5 show some triphone classes that result in improved triphone modelling. The label for each of the synthetic-hybrid and baseline model comparison experiments indicates how many state-specific mixture components we used for the synthetic-hybrid model and the baseline model of the matching label. Using this strategy, Figure 8.5 shows that the difference in log likelihood yields a greater improvement for the "Hybrid : 1 Mix" experiment than for the "Hybrid : 2 Mix" or "Hybrid : 4 Mix" experiments. Clearly, it is more difficult to improve further results for the most specialised models where four mixtures are already employed. In the "Hybrid : 4 Mix" experiment, we found that some synthetic-hybrid triphone models still provided a closer match to the development data than the baseline models of the same triphone labels. We performed the untied model analysis only up to a maximum of four mixtures, owing to the limited number of triphone examples available in the training data. The maximum of only 15 training examples is still limited compared to the number of phone examples used for training the tied-triphone classes in standard ASR systems.

Figures 8.7 and 8.8 show results that are a closer comparison to standard tied-triphone systems. In this case, we experimented with example-selected HMMs for tied-triphone classes. Figure 8.7 compares the mean difference in log likelihood between single-mixture and eight-mixture tied HMMs for each triphone label in the sparse triphone set. As for the untied model analysis descibed above, we then analyse the effect of including synthetic examples for the model estimation of the sparse triphone label set and plot this result in Figure 8.8.

As stated in Section 8.4.1, the state-clustering process ensures that sufficient phone examples of closely related phones are modelled together. Therefore, eight mixture

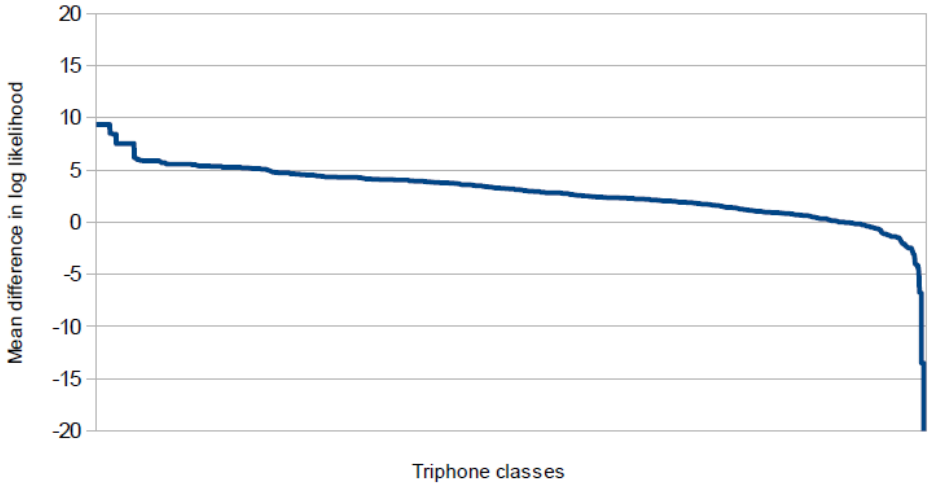

FIGURE 8.7: *Comparing the improvement in mean log likelihood (γ values) between, the tied HMMs of baseline single mixture and baseline eight mixtures. Most, but not all, of the triphones show improved values for eight mixture HMMs.*

models could be examined for these experiments, given the amount of training data. Figures 8.7 and 8.8 plot the mean difference in log likelihood on a per-triphone label basis. These results may show that more than one triphone label has exactly the same value. If all the states of a particular triphone label map to the same tied clusters of another label, the two labels are equivalent.

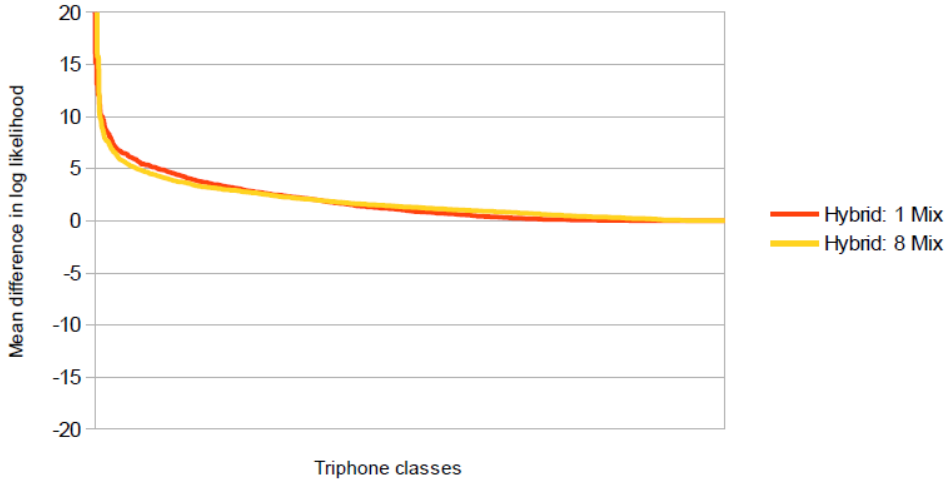

FIGURE 8.8: *Comparing the difference in mean log likelihood of the sparse triphone set for tied HMMs with one- and eight-mixture components per state shows improved likelihoods could be obtained for a large number of triphone classes (94%).*

Again, the first experiment (Figure 8.7) showed the improvement in log likelihood between single-mixture baseline models and baseline models employing eight mixtures. It is interesting to note that not all of the development set examples of the sparse

triphone set showed improvement when using the eight-mixture models. Using eight-mixture models did provide adequate gains for most of the class labels, which is why mixture incrementing is an established approach to improving acoustic models. The main observation made for the other two experiments ("Hybrid : 1 Mix" and "Hybrid : 8 Mix") in Figure 8.8 was that improved likelihoods could be obtained for a large number of triphone classes (94%). Selecting the specific tied-triphone clusters (given all the different synthetic-hybrid HMMs) yielded comparable gains, as observed for the mixture-incrementing strategy employed in standard ASR systems.

8.6.2 Test data

The improved likelihood result given the development data (Figure 8.8) for tied-triphone model selection is encouraging. To determine whether these results generalise to unseen test data, we repeated the same analysis on the test data set, but without re-selecting models. For simplicity, we evaluated only phone labels of the sparse triphones set. These labels also occur in the development data. We carried over the same selection of synthetic triphone examples (obtained by analysing the development data), re-evaluating the selected example-selected HMMs. Lastly, we decided to evaluate the eight-mixture models only.

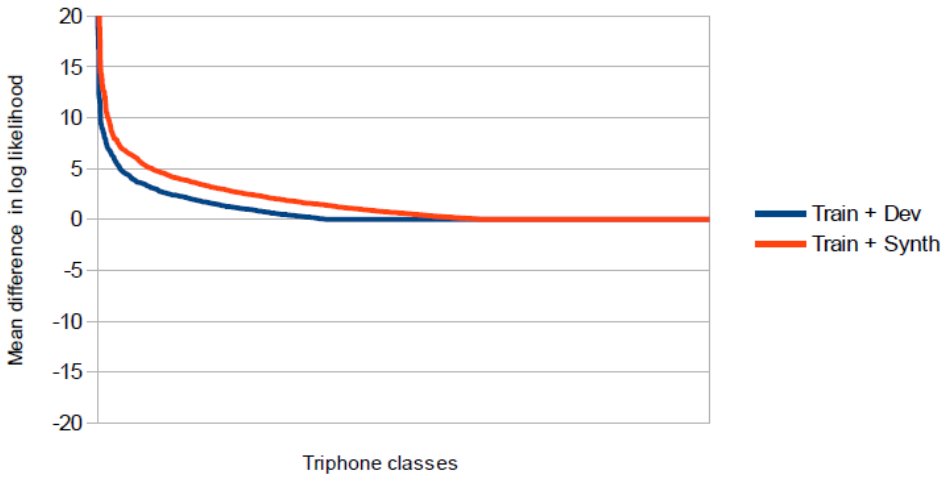

FIGURE 8.9: *Comparing the improvement in mean log likelihood (γ values) of sparse triphone labels for tied HMMs on test data. Training on both the development set and $train_{30}$ data (Train + Dev) still does not outperform the previously trained synthetic-hybrid models (Train + Synth).*

Figure 8.9 depicts two results. As described above, the first "Train + Synth" experiment, repeats the analysis of the "Hybrid : 8 Mix" result shown in Figure 8.8, but evaluates the chosen triphone examples of the test data set. Also, the graph of the "Train + Dev" experiment depicts an analysis of exactly the same test examples, but in this

case a different set of training data was used for model estimation (described below). According to the plot, the "Train + Synth" experiment still shows much improvement for a large number of triphone classes with comparable magnitude to the results observed for the development data.

In the case of real-world systems and especially when the available training data is limited, it would not be desirable to have a separate development data set that could not be used for training purposes. The last experiment "Train + Dev" tested what happened if one trained a set of models, using all the data of the $train_{30}$ and the development set combined. Would this new baseline trained on more data outperform all the previously trained synthetic-hybrid models? The result shown in Figure 8.9 indicates less improvement for the now more resource-constrained synthetic-hybrid models, but for some triphones the previous synthetic-hybrid models still outperform the better-resourced baseline.

8.7 Augmenting ASR systems

The previous section showed that synthetic triphones improved the modelling ability of HMMs. Determining the parameters to incorporate these samples into an ASR system would require further study. We therefore obtained initial results using a simple pooling method, as a final demonstration of the utility of the trajectory analysis. Table 8.4 presents these phone recognition accuracies for test systems. As before, the table displays recognition results before (ACC) and after semi-tied transforms (ACC semi-tied). We augmented the training data with synthetic triphone examples from the sparse triphone set. Examples of triphone labels for which the mean difference in log likelihood has a positive value in Figure 8.8 were added. The total number of selected synthetic triphone examples was 5 526.

System	**ACC**	**ACC (semi-tied)**
Linear	85.21	83.81
Synthetic linear	85.69	86.25
Standard MFCCs	86.94	87.04
Synthetic standard MFCCs	86.94	88.30
Augment dev data	88.88	89.78
Synthetic augment dev data	89.35	90.16
Dev: Linear	85.50	83.62
Dev: Synthetic Linear	85.91	86.61
Dev: Standard MFCCs	87.15	87.09
Dev: Synthetic Standard MFCCs	87.34	87.76

TABLE 8.4: *Comparing phone recognition results for systems trained with and without synthetic training data and adjusting insertion penalties on the development data (Dev).*

We experimented with three main types of acoustic models trained on different feature sets. Firstly, specialised MFCCs (Linear and Synthetic linear systems) could be used for model training (similar to the ASR systems in Section 6.5.2). These systems made it possible to evaluate the effect of "smoothed" trajectory-based training features on the recognition results of the current analysis. Employing "Standard MFCCs" as in normal ASR systems (no additional feature smoothing) tests, whether adding synthetic examples to such a system, can make a difference. Apart from these tests, Table 8.4 includes one additional experiment. Similar to the experiment depicted in Figure 8.9, the "Augment dev data" and "Synthetic augment dev data" systems evaluate whether adding synthetic triphones after including the development data with the $train_{30}$ train data still leads to improvement.

As before, we adjusted the insertion penalty using the development data (Dev). The bottom part of Table 8.4 presents the phone recognition results for these development set tests. For the "Augment dev data" tests, we re-used the insertion penalty obtained for the "Standard MFCCs" system.

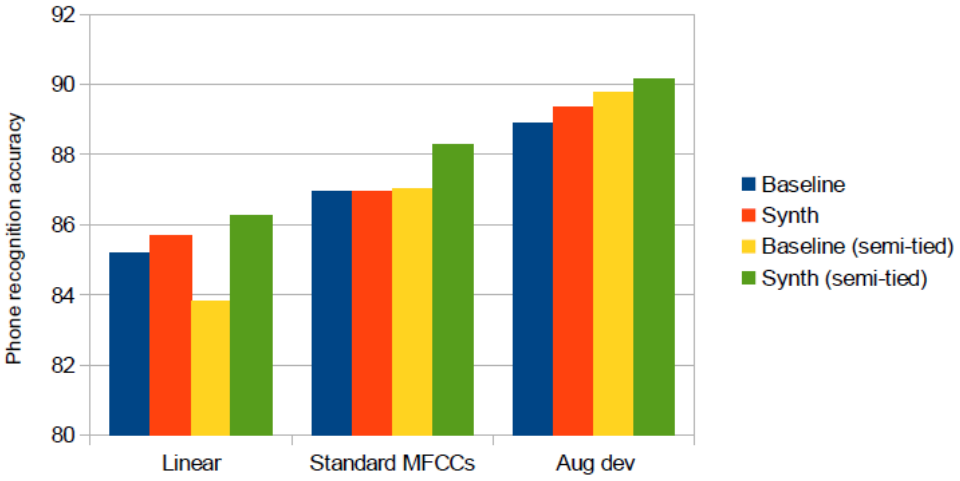

FIGURE 8.10: *Comparing phone recognition results for systems trained with and without synthetic training data. Adding synthetic examples improves system performance.*

Figure 8.10 depicts results from the upper part of Table 8.4. The results of three system types (Linear, Standard MFCCs and Aug dev) are grouped together in groups of four bars each. The blue bars (Baseline) represent each "ACC" result of systems that implement no data augmentation. An orange bar (Synth) displays the results when data augmentation is applied. Similarly, the yellow and green bars (Baseline semi-tied and Synth semi-tied) depict the accuracies when also applying semi-tied transforms.

The "Linear" system results in Figure 8.10 show that using trajectory-based ASR features to represent the limited $train_{30}$ data and testing trajectory-based ASR features

(estimated with oracle alignments) performed worst. Note that oracle alignments only apply to the "Linear" systems results, but for the "Standard MFCCs" and "Aug dev" systems we performed true phone recognition. Employing a semi-tied transform lowers the accuracy achieved even further (83.81%). The semi-tied transform simply does not operate well, given the limited data combined with the smaller variances of smoothed trajectory features. Adding the synthetic triphone examples (selected using the "Hybrid: 8 Mix" experiment in Figure 8.8) led to improvement. In fact, the additional training examples seem to be sufficient to restore the correct functioning of the semi-tied transform (the synthetic linear system achieved 86.25% accuracy).

The performance of the "Linear" system experiments was lower than using standard MFCC features for the $train_{30}$ and test data sets. The phone recognition accuracies of these systems were 86.94% and 87.07% using a semi-tied transform. Although the synthetic triphone examples are trajectory-based ASR features, we tested whether including these examples allow a standard ASR system to incorporate the additional information successfully. Perhaps the standard features of the $train_{30}$ would be sufficient to account for the additional model variance required. The "Synth" experiment found that simply adding the synthetic triphone examples used in this study to the training data does not degrade system performance. Using a semi-tied transform obtained absolute system improvement of 1.26%. Finally, the results in Figure 8.10 conclude that it is possible to achieve improved system performance by including the same synthetic triphone examples with a set of training data of about 50 minutes in duration (Aug dev). Since the synthetic examples model only information of the $train_{30}$ data set, even better results can be expected when creating the synthetic triphones using all the training data ($train_{30}$ combined with the development data). Nonetheless, adding the synthetic examples still improved system performance from 89.78% to a phone accuracy of 90.16% during this test.

8.7.1 Monophone analysis

A phone confusion analysis of the phone recognition system results presented in Table 8.4 and Figure 8.10 is an alternative way to identify phone labels where the selected synthetic triphones are most effective. Such a selection does, however, suffer from the same limitations as the tied-state clustering approach when adding synthetic triphones. Selecting for particular triphone labels interfere with the model estimation of other triphones during context clustering (see discussion in Section 8.8). As a further investigation and to verify system behaviour, we report on the monophone identities where phone recognition achieved improved correctness. We further analysed only the "Standard MFCCs" and "Synthetic standard MFCCs" experiment pair as labelled in Table

8.4. Semi-tied transforms were included, so that the "Baseline (semi-tied)" and "Synth (semi-tied)" result in Figure 8.10 for the "Standard MFCCs" system type is the chosen result. To this end, the results in Table 8.5 report on phone identities where recognition of either the development data or the test data show increased phone correctness employing synthetic triphones. The table provides two phone lists (Phone), which firstly report on the phone labels with improved correctness, given the development data as a test case (Improved phones: Dev). We conclude this section with one additional ASR experiment where we retained only the synthetic examples of these phone labels (Table 8.6). The second list of phone labels shows the error made by only considering the results of the development set. For these phone labels (Additional improved phones: Tst), the development data analysis did not agree that recognition of the test set would show improved correctness. Lastly, we include the number of recognised examples (#Num rec) for the "Standard MFCCs" system result of each phone label.

Phone	**#Num (rec)**	**Control**	**Synth**	**Diff (Dev)**	**Diff (Tst)**
Improved phones: Dev					
g	6	0.0	33.3	33.3	0.0
w	10	50.0	80.0	30.0	37.8
2:	21	14.3	38.1	23.8	15.8
p	167	76.6	84.6	8.0	6.2
y	47	48.9	56.2	7.3	7.6
A:	304	89.5	94.8	5.3	1.0
9y	65	60.0	65.2	5.2	3.1
u@	177	92.1	95.5	3.4	0.1
{	124	84.7	88.1	3.4	0.9
a	292	86.3	89.1	2.8	3.3
t	650	94.8	97.3	2.5	4.2
i@	232	96.1	97.0	0.9	-3.6
r	791	98.1	98.5	0.4	0.6
n	609	96.4	96.6	0.2	-1.5
Additional improved phones: Tst					
@u	31	80.6	87.1	0.0	6.5
E	140	76.4	81.1	-2.9	4.7
h\	37	56.8	58.3	-12.0	1.5
@	1396	96.2	97.3	0.0	1.1
k	402	96.0	96.6	0.0	0.6
l	460	96.7	97.2	-0.1	0.5
s	753	99.5	99.7	0.0	0.2

TABLE 8.5: *The effect of adding synthetic examples to training data on phone correctness, comparing the synthetic and control systems. Most monophones that show improved correctness for the development data also show improved correctness in the test data (Improved phones: Dev). For a list of seven phone labels the test data show improved phone correctness and the development set result does not (Additional improved phones: Tst).*

Considering only the improved phone labels for the development data (Table 8.5), one sees that the synthetic system recognised more examples correctly than the standard system for 14 monophone classes. Accordingly, we report positive values for the difference between the correctness values. Sorting the list of phone identities according

to these values, one finds that the phone identities showing most improvement do not necessarily display the highest number of examples recognised. Consequently, we expect the other phones classes with smaller gains to contribute significantly to the overall phone accuracy. Moreover, the agreement of the correctness results of the test set with the findings of the development data supports the previous finding that selecting phone classes given the development data generalises well to results for the test data set.

There are, however, a list of seven phone labels for which the test data shows improved phone correctness and the development set result does not. Development set results for most of these phones do not degrade either. This finding agrees with the synthetic triphone selection strategy based on development data in the previous section.

Further restricting the choice of synthetic triphones to include only examples mapping to the 14 monophone labels that show improved correctness for the development data and re-evaluating the "synthetic" ASR system gave the result in Table 8.6.

System	**ACC**	**ACC (semi-tied)**
Standard MFCCs	86.94	87.04
Synthetic standard MFCCs	87.40	87.95
Dev: Standard MFCCs	87.15	87.09
Dev: Synthetic standard MFCCs	87.17	87.54

TABLE 8.6: *Restricting synthetic training examples to monophone classes that improve the development set results still produced improved recognition results, but not to the extent of using all the synthetic examples chosen by the likelihood analysis.*

As expected, adding fewer synthetic examples still improved phone recognition accuracies (see Synthetic standard MFCCs in Table 8.6), but not to the extent of using the complete set that was selected acoustically (reported on in Table 8.4 and Figure 8.10). It is clear that using all of the synthetic examples chosen by the likelihood analysis performs best.

8.8 Discussion

This chapter presented an acoustic model analysis to show that synthetic training examples can improve model estimation. We simulated complete phone units that allow the additional training features to be included as synthetic utterances. The histogram in Figure 8.4 shows the sparse nature of the distribution of triphone examples that is typical of 30 minutes of training data. This sparseness allows ample room to experiment with rare and unseen contexts: the focus of this work. We were able to demonstrate that the process of synthesising additional triphones from the trajectories of diphone and monophone units produced synthetic triphone units that could directly improve

data sufficiency, by improving the likelihood of unseen test data within an HMM-based system.

However, as mentioned in Section 8.1, some of the synthetic samples may be of poor quality. When experimenting with untied models a large portion of triphone labels show exactly this. Adding synthetic examples for these phone labels does not provide improved likelihoods. Also, fewer triphone labels show improved estimation for more specific models (compare four-mixture models with one- and two-mixture models). We expected that using larger numbers of mixtures would require more examples for adequate estimation. However, we assumed that triphones that do not show any improvement at 15 synthetic examples would not improve with more synthetic data. The second part of Section 8.6.1 shows a solution to this problem. Sharing information among related groups of triphones improves the applicability of synthetic examples considerably.

The estimation of tied-triphone models creates tied states that group together 30, even up to 60 closely related phone examples. In a similar manner, the synthetic triphone examples created by the technique used in this study should relate (as closely as possible) to real phone examples. The tied-triphone model results demonstrate that it is indeed possible to select tied states by controlling the amount of synthetic data and other closely related triphone classes. Such a mechanism provides an improved match of the test data. Only when more data is shared using other real phone examples do most triphone labels benefit from synthetic examples (Figure 8.8). This process also seems to be more stable, in the sense that the gains observed remain similar whether the acoustic models are trained using one or eight mixtures. The standard decision tree clustering process implemented in HTK [92] ensures model estimation using a sufficient number of examples, while the likelihood analysis and selection strategy allow acoustic accuracy to be maintained. Together these two aspects can be controlled, providing a new mechanism through which data sufficiency in speech systems may be improved.

The experiments in Section 8.6.2 verified that the likelihood-based selection strategy generalises across different test data sets. Moreover, it is also possible to drive this selection process directly from the available training data. Tied-state clustering seems to be effective in sharing sufficient information among closely related triphone classes adequately constraining the selection process of synthetic examples.

8.9 Conclusion

This chapter demonstrated that it is possible to build complete phone examples with larger contexts from the trajectories of smaller sub-phone units. For some rare and

unseen contexts the synthetic examples provided improved estimation of acoustic models. Since some synthetic examples lack the quality of real examples, data sharing strategies are of interest. It was found that the tied-state modelling of current ASR systems can learn additional information from a much bigger pool of synthetic examples.

Chapter 9

Conclusion

9.1 Introduction

The discussion in Section 1.5 introduced the main research aims we investigated in this thesis. From literature it was clear that inadequate context modelling, which occurs especially with limited speech data, deteriorates the accuracy of speech systems. In our view, data sharing (on a contextual level) is a necessary development to improve the acoustic representation of under-resourced systems. To make this type of data sharing as practical as possible, we designed a simple piecewise linear representation of feature trajectories that closely follow the spectra. Trajectories based on spectral information can be simplified to the extent that a linear representation is adequate. Designing models that capture the trajectory information allows simulation of additional training examples. Critically, this work addresses the issue of whether these simulated examples can successfully augment training data and consequently improve model estimation for speech processing systems. A summary of contributions follows and then future work is discussed.

9.2 Summary of contribution

The complex way in which co-articulation modifies speech features would require very precise transformations during acoustic modelling. This thesis was able to demonstrate an alternative approach to address this issue. Simulation of many unrepresented contexts can be achieved using synthetic speech data and does not require further model refinement. Synthetic phones made from sub-phone trajectory units can alleviate data sufficiency constraints and improve the likelihood of test data. The main contributions made during this research include the following:

- New techniques to track co-articulation effects for frame-based features. Subsequent analysis shows the captured information from phone transitions to be both informative and able to predict the characteristics of particular types of transitions [97].

- Development and analysis of a new high-quality speech corpus for trajectory tracking (the ATTC). In a period of 3 hours and 20 minutes, this corpus contains more high-quality diphone examples for a single speaker than those usually found in speech corpora. The recordings were made for short utterances (three to five words), which means data set selection can be done on a per-utterance basis to easily include a required set of diphones [98].

- The development of a linear piecewise model representation of diphone transitions. These trajectories model the location and magnitude where the features of phones stabilise. Successful analysis of MFCC and spectral (filter bank) features show adequate representation. The model should also be applicable to other speech features that represent phone transitions using spectral information [94, 98].

- Improved trajectory tracking by combining core techniques in noise robustness approaches for speech recognition with a linear representation of spectral features. Removing unwanted high-frequency components from the spectrum allows most of the contextual information (concentrated at lower frequencies) to pass the filtering process unaffected. Chapter 6 demonstrates that this same principle also improves linear trajectory tracking [99].

- A demonstration that tracking trajectories for spectra rather than for cepstra simplifies the trajectory representation (Chapter 6). This makes the simulation of unseen trajectories practical and allows the flexible construction of trajectories from sub-components. Modelling the co-variance among related parameters encodes sufficient spectral information from which to construct ASR features directly.

- Trajectory-based refinements to ensure optimised alignments (Sections 7.6.2 and 7.8) for single transition models (Section 7.5) that successfully integrate into the larger framework.

- Development of a much simpler segment-based trajectory tracking algorithm (Section 7.3.3.2), allowing fast trajectory estimation with comparable quality to a full dynamic programming approach for trajectory tracking (Section 7.8).

- Models (see [99] and Chapter 8) to predict unseen diphone trajectories, synthesising triphone examples for rare and unseen contexts. Adding synthetic training examples for these contexts leads to better acoustic representations. A likelihood

analysis of the new rare and unseen context models reveals that up to 94% of rare and unseen triphones show increased likelihood. This finding may lead to the development of a new mechanism to analyse and improve data sufficiency through context clustering during acoustic model estimation. Although the current method to model the clusters of triphone examples using HTK [92] does not yet include such a mechanism, we were able to demonstrate successful augmentation of training data. By pooling the synthetic examples chosen during the likelihood analysis with the training data, these systems show a small but consistent improvement in recognition accuracy for the data we analyse.

9.3 Future work

Analysis of the current approach demonstrated the feasibility of using context-specific information to model unseen or rare contexts better. This has opened a number of new avenues for research, both in refining the models themselves and in ensuring that their implementation is computationally efficient. Specifically, analysis of more detailed contextual effects now becomes possible: extending the current work to larger units, investigating broad phonemic categories of contexts and analysing the effect of selecting different sub-components (mixing smaller or larger contexts) during synthetic unit construction may all lead to more accurate synthetic units. In addition, future research should focus on robust trajectory estimation for different speakers and recording environments. Enhanced development of low-resource speech systems is the desired outcome of these improved data mining strategies. To discuss these ideas, we focus on the issue of feature mismatch and the importance of using trajectory features as training data:

- One of the critical goals of research in noise-robust speech systems is to reduce model-feature mismatch for models trained in a particular controlled environment and recognition features in the noisy environment. Such techniques are only successful if the information required for recognition is preserved adequately. Cancelling out the differences between environments leads to decreased feature variance that should benefit trajectory-based analysis. Over-smoothing causes difficulty for trajectory modelling to detect phone transitions accurately. The lost transitional information may be recovered. For example, the authors of [16] borrow image processing techniques (such as edge detection) to recover information lost by low-pass filtering.

- Typical ASR systems use features such as MFCCs, PLPs or features derived from them, such as bottleneck features, to encode the speech waveform. As explained

in Chapter 2, Section 2.5.2.1, the authors of [83] believe the advantages of DNN-HMM systems strongly motivate their application to speech recognition in low-resource languages. Further analysis incorporating trajectories as features to the input of such systems is required. A better understanding of contextual effects on the feature level (using trajectory information) could inform further system development to train speech systems with limited resources using discriminative and DNN-HMM solutions.

Improved trajectory-based analysis may lead to a deeper understanding of pronunciation differences and the development of new methods of data sharing. The next three items highlight some immediate aspects that future work may pursue:

- Acoustic modelling approaches rely heavily on tied-state clustering when training data is limited. The difference in quality of synthetic data created by newly emerging augmentation approaches to the acoustic quality of real speech examples introduce new avenues of research. It is not clear exactly how to share the information of synthetic phone examples with real phone examples. The work in Chapter 8 experimented with decision tree clustering, tying HMM states of closely related triphone labels. There may be more appropriate ways to deal with this problem, possibly even at the feature level.

- Recent research perturbed speech data using a VTLP approach (see description in Section 2.5.2.1). The same technique may be used to simulate the trajectories for different speakers, borrowing from an inventory of speaker-specific trajectories.

- A better understanding of the acoustic distances for specific contexts in the cross-speaker and cross-language domain could aid further research. Trajectory modelling may be a very useful way to analyse the properties of this larger acoustic space. In fact, the evaluation techniques used in this thesis to assess the ability of trajectories to predict various phone transitions can be directly applied to compare trajectories of similar contexts between speakers.

- Different speakers of the same language may pronounce and co-articulate phones differently. Section 1.4 introduces the idea that pronunciation may be idiosyncratic. Research to model pronunciation variance and its effect on feature trajectories will be required to improve data sharing among speakers.

Apart from enhanced ASR development, trajectory-based representations should also apply to TTS systems and language learning implementations:

- Computer-assisted language learning approaches could potentially benefit from improved pronunciation modelling. The capability to analyse and more accurately compare pronunciations (using feature trajectories) could benefit evaluation approaches. Improved confidence scoring may be possible using trajectories that encode co-articulation effects explicitly.

- Synthetic training examples could potentially aid the construction of high-quality TTS voices when data is limited. Voice artists have to record phonetically balanced sets of sentences to capture most of the co-articulatory effects accurately. This process is time-consuming and costly (especially if a selection of voices is required). Since the HTS framework utilises HMM-based modelling, synthetic training examples derived from the same studio quality data could also improve estimation of rarely seen contextual units. Alternatively, synthesising speech with unit selection approaches produces high-quality voices only with vast amounts of data from a single speaker. The technique attempts to concatenate the best units. Further research could be undertaken to determine whether synthetic units could provide units that if concatenated, produce fewer artifacts and pronunciation errors for a limited data set.

9.4 Conclusion

This thesis contributed new methods to represent, analyse and apply feature trajectories to increase data sufficiency when speech data is limited. This is of relevance, as many languages still lack HLT resources and can therefore not benefit from high-quality speech processing technology. The trajectory modelling approach described here provides new tools to analyse and understand contextual effects, which could inform future speech modelling approaches in various different domains.

Appendix A

Metric definitions

In this appendix we list definitions of all the metrics used in the thesis.

A.1 Transition model parameters

The purpose of this section is to present the transition model parameters and calculations based on transition models parameters for the models introduced in Section 5.3.1.

Symbol	Calculation	Description
Measured transition model parameters		
Seg_{start}	-	frame number at start of the diphone segment
Seg_{end}	-	frame number at end of the diphone segment
$T1$	-	frame number relative to Seg_{start} at start of the transition
$T2$	-	frame number relative to Seg_{start} at end of the transition
$S1$	-	parameter value at initial stable value
$S2$	-	parameter value of final stable value

TABLE A.1: *Trajectory model parameters describing a single transition.*

Note that the analysis window between Seg_{start} and $Seg_{start} + T1$ is referred to as the "initial stable segment"; between $Seg_{start} + T1$ and $Seg_{start} + T2$ as the "change descriptor", and between $Seg_{start} + T2$ and Seg_{end} as the "final stable segment".

Symbol	Calculation	Description
Calculated transition model parameters		
T_{dur}	$T2 - T1$	duration of the change descriptor
T_{mid}	$\frac{T1+T2}{2}$	centre of the change descriptor
Seg_{dur}	$Seg_{end} - Seg_{start}$	duration of the segment
R_{mid}	$\frac{T_{mid}}{Seg_{dur}}$	centre of the change descriptor, relative to the segment duration
R_{dur}	$\frac{T_{dur}}{Seg_{dur}}$	duration of the change descriptor, relative to the segment duration

TABLE A.2: *Frame-based trajectory model parameters (in number of frames) calculated from the values in Table A.1.*

A.2 Defined variable operations

Definitions of the variable operations such as the Pearson correlation coefficient, Euclidean distance and dot product we used in Chapter 3 and the filtering operations of Chapter 6 are as follows:

Symbol	Calculation	Description
Frame-based feature smoothing		
$M(X_i)$	$\frac{X_{(i-m)}+\ldots+X_{(i-1)}+X_i+\ldots+X_{(i+m)}}{2m+1}$	MA filter of order m, processing an input vector X and i the current element index.
$A(X_i)$	$\frac{R_{(i-m)}+\ldots+R_{(i-1)}+X_i+\ldots+X_{(i+m)}}{2m+1}$ where $R_{(i-m)} = A(X_{(i-m)})$	ARMA filter of order m, processing an input vector X, i the current element index and auto-regressive ARMA post-processed feature vector R.
Distance metrics between random variables		
$r(X,Y)$	$\frac{\sum_{i=1}^{N}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{N}(X_i-\bar{X})^2}\sqrt{\sum_{i=1}^{N}(Y_i-\bar{Y})^2}}$	Pearson correlation coefficient between two random variables X and Y, each with N dimensions. $\bar{X}$ and $\bar{Y}$ are the means of the random variables X and Y respectively
$d(X,Y)$	$\sqrt{\sum_{i=1}^{N}(X_i-Y_i)^2}$	Euclidean distance between two random variables X and Y for N dimensions.
$X \cdot Y$	$\sum_{i=1}^{N} X_i Y_i$	Dot product between two random variables X and Y with N dimensions.

TABLE A.3: *Variable operations.*

A.3 Statistical properties of parameters

A set of statistical properties were used to approximate various parameters thought the work of this thesis. Table A.4 presents a list of these definitions below.

Symbol	Calculation	Description
Statistical properties of parameters		
$\hat{\mu}(p,\omega)$	$\frac{1}{S}\sum_{s=1}^{S} p_s$	The mean of a specific parameter p, estimated over all S samples in set ω.
$\hat{\sigma}(p,\omega)$	$\sqrt{\frac{1}{S}\sum_{s=1}^{S}(p_s-\hat{\mu}(p,\omega))^2}$	The standard deviation of a specific parameter p, estimated over all S samples in a set ω.
$\hat{\sigma}_{corrected}(p,\omega)$	$\sqrt{\frac{1}{S-1}\sum_{s=1}^{S}(p_s-\hat{\mu}(p,\omega))^2}$	The corrected standard deviation of a specific parameter p, estimated over all S samples in a set ω.
$\hat{q}(p,\hat{p},\omega)$	$\frac{1}{S}\sum_{s=1}^{S}(p_s-\mu(p,\omega))(\hat{p}_s-\mu(\hat{p},\omega))$	The covariance of a specific parameter p and its estimate $\hat{p}$, calculated over all S samples in set ω.
$\hat{\epsilon}(p,\omega)$	$\frac{\hat{\sigma}(p,\omega)}{\sqrt{S}}$	Standard error of the mean estimator $\hat{p}(\omega)$, for S samples in a set ω.
$SE(p_s,\hat{p}_s,\omega)$	$(p_s-\hat{p}_s)^2$	Squared error between the actual measured value of a parameter sample p_s and its estimate $\hat{p}_s$, where the sample is drawn from the set ω.
$MSE(p,\hat{p},\omega)$	$\frac{1}{S}\sum_{s=1}^{S}(p_s-\hat{p}_s)^2$	Mean squared error of the estimator $\hat{p}_s$ of a specific parameter p and individual parameter samples p_s, estimated over all S samples in set ω.
$WMSE(p,\hat{p},\omega)$	$\frac{1}{S}\sum_{s=1}^{S}\frac{(p_s-\hat{p}_s)^2}{\hat{\sigma}^2(p,\omega)}$	Variance-weighted mean squared error of the estimator $\hat{p}_s$ of a specific parameter p and individual parameter samples p_s, estimated over all S samples in set ω.
$r(p,\hat{p},\omega)$	$\frac{\hat{q}(p,\hat{p},\omega)}{\hat{\sigma}(p,\omega)\hat{\sigma}(\hat{p},\omega)}$	The Pearson correlation coefficient between a measured parameter p and its estimated value $\hat{p}$ for all samples in the set ω.

TABLE A.4: *Statistical properties that are used to describe and analyse parameters and parameter estimators throughout this work.*

Note that the corrected sample standard deviation $\hat{\sigma}_{corrected}(p,\omega)$ still includes bias due to the square root being a concave function. This bias are still significant for less than 10 examples.

A.4 Model-feature approximation metrics

Trajectory model estimation and assessment required various model-feature approximations. Tables A.5 and A.6 lists these metrics and Table A.7 presents the global approximation metrics.

Symbol	Calculation	Description
Model-feature approximation metrics		
p_f	various	Parameter p at frame f.
$\hat{p}_f$	various	Estimate of a parameter p at frame f.
SE_{frame}	$SE(p_f, \hat{p}_f, \omega)$ where	SE value of a specific frame f.
	ω is $\bigcup_{p_f, \hat{p}_f}$	
$SE_{segment}$	$\sum_{f=1}^{F_{cs}} SE(p_f, \hat{p}_f, \omega)$ where	Sum of the SE values of a single diphone segment and feature channel c, over the set (ω)
	ω is $\bigcup_{p_f, \hat{p}_f} \bigcup_{f=1}^{F_{cs}}$	consisting of all frames of this sample segment s.
$MSE_{trans}(c, d)$	$MSE(p, \hat{p}, \omega_{cd})$ where	MSE of a single feature channel c, and the diphone transition d, over the set (ω)
	ω_{cd} is $\bigcup_{p_{fc}, \hat{p}_{fc}} \bigcup_{s=1}^{S_d} \bigcup_{f=1}^{F_s}$	consisting of all frames of all samples S_d that contains this diphone.
$MSE_{channel}(c)$	$MSE(p, \hat{p}, \omega_c)$ where	MSE of a feature channel c, over the set (ω)
	ω_c is $\bigcup_{p_{fc}, \hat{p}_{fc}} \bigcup_{f=1}^{F_c}$	consisting of all frames F_c.
$MSE_{diphone}(d)$	$MSE(p, \hat{p}, \omega_d)$ where	MSE of a diphone transition d, over the set (ω) consisting of all frames of all samples S_d
	ω is $\bigcup_{p_f, \hat{p}_f} \bigcup_{s=1}^{S_d} \bigcup_{c=1}^{C} \bigcup_{f=1}^{F_{cs}}$	that contains this diphone, summed over all feature channels C.
$r_{trans}(c, d)$	$r(p, \hat{p}, \omega_{cd})$ where	Pearson correlation for a single feature channel c and diphone d, over the set (ω)
	ω_{cd} is $\bigcup_{p_{fc}, \hat{p}_{fc}} \bigcup_{s=1}^{S_d} \bigcup_{f=1}^{F_s}$	consisting of all frames of all samples S_d that contains this diphone.
$r_{channel}(c)$	$r(p, \hat{p}, \omega_c)$ where	Pearson correlation over the set (ω) consisting of
	ω_c is $\bigcup_{p_{fc}, \hat{p}_{fc}} \bigcup_{f=1}^{F_c}$	all frames F_c of a feature channel c.
$r_{diphone}(d)$	$\frac{1}{C} \sum_{c=1}^{C} r_{trans}(c, d)$	Pearson correlation of the diphone transition d, summed over all feature channels C.

TABLE A.5: *Model-feature approximation measurements.*

Symbol	Calculation	Description
Variance-weighted approximation metrics		
$WMSE_{trans}(c,d)$	$\frac{MSE(p,\hat{p},\omega_{cd})}{\sigma^2(p,\omega_c)}$ where	Variance-weighted MSE of a single feature channel c, and the diphone transition d,
	ω_{cd} is $\bigcup_{p_{fc},\hat{p}_{fc}} \bigcup_{s=1}^{S_d} \bigcup_{f=1}^{F_s}$	over the set (ω_{cd}) consisting of all frames
	and ω_c is $\bigcup_{p_{fc},\hat{p}_{fc}} \bigcup_{f=1}^{F_c}$	of all samples S_d that contains this diphone.
$WMSE_{channel}(c)$	$\frac{MSE(p,\hat{p},\omega_c)}{\sigma^2(p,\omega_c)}$ where	Variance-weighted MSE of a feature channel c,
	ω_c is $\bigcup_{p_{fc},\hat{p}_{fc}} \bigcup_{f=1}^{F_c}$	over the set (ω_c) consisting of all frames F_c.
$WMSE_{diphone}(d)$	$\frac{1}{\sum_{s=1}^{S} CF_s} \sum_{s=1}^{S_d} \sum_{c=1}^{C} \sum_{f=1}^{F_s} \frac{SE(p_f,\hat{p}_f,\omega_{cd})}{\sigma^2(p,\omega_c)}$ where	MSE of a diphone transition d, over the set (ω_{cd})
	ω_c is $\bigcup_{p_{fc},\hat{p}_{fc}} \bigcup_{f=1}^{F_c}$	consisting of all frames of all samples S_d
	and ω_{cd} is $\bigcup_{p_{fc},\hat{p}_{fc}} \bigcup_{s=1}^{S_d} \bigcup_{f=1}^{F_s}$	that contains this diphone, summed over all feature channels C.
Parameter estimator error metrics		
p_s	various	Value of parameter p for the sample s.
$\hat{\sigma}_{trans}(c,d)$	$\hat{\sigma}(p_s,\omega)$ where	Consistency of a parameter p for a single feature channel c, and the diphone transition d,
	ω is $\bigcup_{p_s} \bigcup_{s=1}^{S_{cd}}$	over the set (ω) consisting of the value for the parameter p of all samples S_{cd} that contains this diphone in the feature channel c.
$\hat{\epsilon}_{trans}(c,d)$	$\hat{\epsilon}(p_s,\omega)$ where	Standard error of the mean estimator $\hat{p}$ for a single feature channel c, and the diphone transition d, over the set (ω) consisting of the value
	ω is $\bigcup_{p_s} \bigcup_{s=1}^{S_{cd}}$	for the parameter p of all samples S_{cd} that contains this diphone in the feature channel c.

TABLE A.6: *Model-feature approximation measurements.*

Symbol	Calculation	Description
Global model-feature approximation metrics		
$GMSE_{trans}$	$\frac{1}{CD}\sum_{c=1}^{C}\sum_{d=1}^{D} MSE_{trans}(c,d)$	The mean (global) MSE value of all transitions in the data set, over all diphones D and feature channels C.
$GMSE_{channel}$	$\frac{1}{C}\sum_{c=1}^{C} MSE_{channel}(c)$	The mean (global) MSE value for the C feature channels of the data set.
$GMSE_{diphone}$	$\frac{1}{D}\sum_{d=1}^{D} MSE_{diphone}(d)$	The mean (global) MSE value over all diphone labels D in a data set.
$Gr_{channel}$	$\frac{1}{C}\sum_{c=1}^{C} r_{channel}(c)$	The mean (global) correlation value over all C feature channels of the data set.
Global variance-weighted approximation metrics		
$GWMSE_{trans}$	$\frac{1}{CD}\sum_{c=1}^{C}\sum_{d=1}^{D} WMSE_{trans}(c,d)$	The mean (global) MSE value of all transitions in the data set, over all diphones D and feature channels C.
$GWMSE_{channel}$	$\frac{1}{C}\sum_{c=1}^{C} WMSE_{channel}(c)$	The mean (global) MSE value for the C feature channels of the data set.
$GWMSE_{diphone}$	$\frac{1}{D}\sum_{d=1}^{D} WMSE_{diphone}(d)$	The mean (global) MSE value over all diphone labels D in a data set.
Global parameter estimator error metrics		
$G\hat{\sigma}_{trans}$	$\frac{1}{CD}\sum_{c=1}^{C}\sum_{d=1}^{D} \hat{\sigma}_{trans}(c,d)$	The mean (global) consistency value of all transitions in the data set, diphones D and feature channels C.
$G\hat{\sigma}_{diphone}$	$\frac{1}{C}\sum_{c=1}^{C} \hat{\epsilon}_{trans}(c,d)$	The mean (global) consistency value of all transitions of a particular diphone d in the data set, over all feature channels C.
$G\hat{\epsilon}_{trans}$	$\frac{1}{CD}\sum_{c=1}^{C}\sum_{d=1}^{D} \hat{\epsilon}_{trans}(c,d)$	The mean (global) standard error value of all transitions in the data set, over all diphones D and feature channels C.
$G\hat{\epsilon}_{diphone}$	$\frac{1}{C}\sum_{c=1}^{C} \hat{\epsilon}_{trans}(c,d)$	The mean (global) standard error value of all transitions of a particular diphone d in the data set, over all feature channels C.

TABLE A.7: *Global metrics to analyse the model-feature approximation of a data set.*

Appendix B

Algorithms used

This appendix provides detailed pseudo-code for all the algorithms used for creating and testing the trajectory models.

B.1 Estimating free trajectory models

The construction of a free-trajectory model for an utterance has three steps. In the first step, the utterance is segmented into diphone transitions. In the second step (model alignment), a search was made for the $T1$ and $T2$ parameter alignments (see Section 5.3.1) of every segment. Finally, these model alignments allowed the connection of all the segments of an utterance to form the complete trajectory.

B.1.1 Model alignment

Two algorithms were used for obtaining the model alignments. The "AlignSegment" algorithm works purely on a per-segment basis (not taking into account the frames of the previous or next segments). This constraint can be adjusted and the size of the search dynamically adjusted by using a second dynamic programming algorithm "DPSegment".

The variables and functions required by the first "AlignSegment" algorithm (Algorithm 1) is as follows:

- $Feat$: the feature values (one for each frame) for a single channel and diphone segment.
- Length: the function that returns the number of elements of a data structure.

- FirstOrderFit: the function that perform a least square error fit for a first order polynomial function. A list of feature values and two scalar time alignment values is required as input. Then the function returns a single scalar number.

- SE: the function for computing the squared error. Similar to the FirstOrderFit function, two time-alignment values and a list of feature values are needed. In addition, a polynomial line piece (list of coefficients) is required as the feature estimator, with which the error is computed. Then the squared error itself is returned as a single scalar value.

- DrawLine: the function that connects two Euclidean co-ordinates (in two-dimensional space), using a first-order line piece. Two lists, each containing the scalar co-ordinate input values, are required as input. The output is given as a list of polynomial coefficients, describing the first-order line.

- Min: the function that returns the minimum value of a sequence of values. If two minimum numbers are identical, the function returns the first value in the input of these two.

Algorithm 1 Transition model alignments

function AlignSegment($Feat$)
  $Seg_{end} \leftarrow Length(Feat)$
  for $T1 = 2$ to $Seg_{end} - 2$ **do** // ***Min length of 1 for both stable values***
    for $T2 = T1 + 1$ to $Seg_{end} - 1$ **do** // ***Change descriptor min length 0***
      $SE_{total} \leftarrow 0$
      $S1 \leftarrow FirstOrderFit(Feat, 1, T1)$
      $SE_{total} \leftarrow SE_{total} + SE(S1, Feat, 1, T1)$

      $S2 \leftarrow FirstOrderFit(Feat, T2, Seg_{end})$
      $SE_{total} \leftarrow SE_{total} + SE(S2, Feat, T2, Seg_{end})$

      $Change \leftarrow DrawLine([T1, S1], [T2, S2])$
      if $T2 > T1 + 1$ **then**
        $SE_{total} \leftarrow SE_{total} + SE(Change, Feat, T1 + 1, T2 - 1)$
      end if
      $MSE_{trans}(T1, T2) \leftarrow SE_{total}/(Seg_{end})$
    end for
  end for
  $T1, T2 \leftarrow Min(MSE_{trans})$
return $T1, T2$
end function

Given a single stream of feature values for a single diphone example, Algorithm 1 simply returns a set of model alignment values ($T1$ and $T2$). The input used for Algorithm 4 is more closely related to the units obtained directly from ASR segmentation. A single

stream of feature values for the whole sequence of monophones of a whole utterance is provided as input and then a list of all the model alignment values ($T1$ and $T2$) is returned. Algorithms 2 and 3, give additional descriptions of exactly how all the options are added (on a per stable value basis) and the cost of each path is calculated. These calculations allow the selection of a best alignment path and consequently the final list of model alignments can be computed in Algorithm 4.

The detailed description of Algorithm 2 are as follows:

- s: Index of the current stable segment.
- $s1$: Index of alignment ending first stable value of a diphone transition.
- $sync_{list}$: List of index values (one for every feature value), associating each stable segment index with all of its feature values.
- $next_{stable}$: A data structure keeping track of all the model alignment options generated for the second stable value of a diphone transition.
- $Slack$: A variable to adjust how strictly the alignments have to follow the ASR segmentation. Values greater than zero increase the number of feature values that form part of every local search.
- MaxSegIndex: Returns the last feature value index of the stable segment s. If Slack is greater than zero, this index value results from stable segments later than s.
- Last: Returns the last value in a list.
- LastIndex: Returns the index of the last value in a list.

Algorithm 2 Get all stable value options for the next stable segment

```
1: function GetStableOptions(s,s1,sync_list)
2:     for s2 = s1 + 1 to MaxSegIndex(s, sync_list, Slack) do
3:         if sync_list[s2] >= s − Slack then //  Generate options
4:             if s = Last(sync_list) then //  Last stable segment
5:                 s3 ← LastIndex(sync_list)
6:                 next_stable[s2][s3] ← True
7:             else
8:                 for s3 = s2 to MaxSegIndex(s, sync_list, Slack) do
9:                     if sync_list[s3] >= s − Slack then
10:                        next_stable[s2][s3] ← True //  Valid alignment option
11:                    end if
12:                end for
13:            end if
14:        end if
15:    end for
16: return next_stable
17: end function
```

The detailed description of Algorithm 3 are as follows:

- *cost*: Data structure keeping track of the cost associated with the trajectory model of a particular path.
- s: Index of the current stable segment.
- $s0$: Index of alignment starting the first stable value of a diphone transition.
- $s1$: Index of alignment ending the first stable value of a diphone transition.
- $next_{stable}$: A data structure keeping track of all the model alignment options generated for the second stable value of a diphone transition.
- $feat_{list}$: All feature values for this channel of the utterance.
- $MaxCost$: A high-cost value used for initialising the cost of new paths.
- P_{costs}: List of alignment options and their associated costs.
- Keys: Function returning the indices of a data structure.
- MaxSegIndex: Returns the last feature value index of the stable segment s. If Slack is greater than zero, this index value results from stable segments later than s.
- Last: Returns the last value in a list.
- LastIndex: Returns the index of the last value in a list.

- Length: Function that returns the number of elements of a data structure.
- TrackCost: Function generating the trajectory for the associated feature values in $feat_{list}$ between two time alignments. The tracking of the generated trajectory segment is then evaluated and returned as a cost. The cost function employed is simply the sum of the absolute difference between the trajectory value and the feature values.

Algorithm 3 Calculate the cost of valid paths

```
1: function GetPathCosts(cost,s,s0,s1,next_stable,feat_list)
2:     s2_list ← Keys(next_stable)
3:     for s2 = 1 to Length(s2_list) do
4:         s3_list ← Keys(next_stable[s2])
5:         for s3 = 1 to Length(s3_list) do
6:             if not exits cost[s][s2][s3] then
7:                 cost[s][s2][s3] ← MaxCost //  Initialise
8:             end if
9:             PathCost ← cost[s − 1][s0][s1] //  Track and compute updated cost
10:            PathCost ← PathCost + TrackCost(s1 + 1, s2 − 1, feat_list)
11:            PathCost ← PathCost + TrackCost(s2, s3, feat_list)
12:            add (s2, s3, PathCost) to P_costs //  Cost of new path
13:        end for
14:    end for
15: return P_costs, cost
16: end function
```

A detailed description of the model alignment algorithm, based on a dynamic programming approach (Algorithm 4) is now given:

- $Mono_{Feat}$: The feature values (one for each frame) of each monophone and a single channel of a whole utterance.
- $sync_{list}$: List of index values (one for every feature value), associating each stable segment index with all of its feature values.
- *Slack*: A variable to adjust how strictly the alignments have to follow the ASR segmentation. Values greater than zero increase the number of feature values that form part of every local search.
- *cost*: Data structure keeping track of the cost associated with the trajectory model of a particular path.
- *align*: Data structure to store model alignments for different alignment paths for the whole utterance.
- s: Index of the current stable segment.

- $s0$: Index of alignment starting the first stable value of a diphone transition.
- $s1$: Index of alignment ending the first stable value of a diphone transition.
- $next_{stable}$: A data structure keeping track of all the model alignment options generated for the second stable value of a diphone transition.
- $feat_{list}$: All feature values for this channel of the utterance.
- $MaxCost$: A high-cost value used for initialising the cost of new paths.
- P_{costs}: List of alignment options and their associated costs.
- $Stable_{options}$: Data structure that keeps the valid alignment options generated for the previous stable segment.
- $keep_{options}$: Accumulates the newly chosen $Stable_{options}$, until this stable segment can be completed.
- $best_{s2}$: Stores the best alignment for the final stable segment.
- min_{cost}: Stores the minimised cost of the final stable segment choices.
- Seg_{Align}: A list of all the final model alignments ($T1$, $T2$) for this utterance.
- FrameSync: Function generating a list of index values (one for every feature value), associating each stable segment index with all of its feature values.
- InitStableCost: Function generating the trajectory for the first stable segment of feature values in $feat_{list}$ ending at a particular index. Then the tracking of the generated trajectory segment is evaluated and returned as a cost. The cost function employed is simply the sum of the absolute difference between the trajectory value and the feature values.
- Last: Returns the last value in a list.
- Keys: Function returning the indices of a data structure.
- MaxSegIndex: Returns the last feature value index of the stable segment s. If Slack is greater than zero, this index value results from stable segments later than s.
- Length: Function that returns the number of elements of a data structure.

Algorithm 4 Dynamic programming approach to transition model alignments

```
1: function DPSegment(Mono_feat)
2:     for i = 1 to Length(Mono_feat) do // Associate frames with monophones
3:         sync_list ← FrameSync(Mono_feat)
4:         add Mono_feat[i] to feat_list // Features of complete channel
5:     end for
6:
7:     for f = 1 to MaxSegIndex(1, sync_list, Slack) do
8:         cost[1][1][f] ← InitStableCost(feat_list,f)
9:         add (1, f) to align[1][1][f] // Store alignments for paths
10:        Stable_options[1][f] ← True // Identify valid alignment option
11:    end for
12:
13:    for s = 1 to Last(sync_list) do // One stable segment at a time
14:        s0_list ← Keys(Stable_options)
15:        for s0 = 1 to Length(s0_list) do
16:            s1_list ← Keys(Stable_options[s0])
17:            for s1 = 1 to Length(s1_list) do // First stable value indexes
18:                next_stable ← GetStableOptions(s,s1,sync_list)
19:                P_costs, cost ← GetPathCosts(cost,s,s0,s1,next_stable,feat_list)
20:                (s2, s3, cost[s][s2][s3]) ← Min(P_costs) // Record best path
21:                align[s][s2][s3] ← align[s − 1][s0][s1]
22:                add (s2, s3) to align[s][s2][s3]
23:                keep_options[s2][s3] ← True // Add best option for this (s0,s1)
24:            end for
25:        end for
26:        Stable_options ← keep_options
27:    end for
28:
29:    min_cost ← MaxCost // Find best path with last stable segment
30:    best_s2 ← −1
31:    for each s2 in Keys(cost[Last(sync_list)]) do
32:        if exists cost[Last(sync_list)][s2][Length(feat_list)] and < min_cost then
33:            min_cost ← cost[Last(sync_list)][s2][Length(feat_list)]
34:            best_s2 ← s2
35:        end if
36:    end for
37:
38:    if best_s2 = −1 then
39:        Exit Algorithm // No valid alignment path found
40:    else
41:        SegAlign ← align[Last(sync_list)][best_s2][Length(feat_list)]
42:    end if
43: return SegAlign // A list with all (T1,T2) alignments
44: end function
```

B.1.2 Connecting segments

This section describes the variables and functions required by the algorithm connecting a sequence of transitions (Algorithm 5). The purpose of this algorithm is to leave the model alignments unchanged, but to update stable values to produce connected trajectories:

- Seg_{Feat}: All the diphone segments of a single feature channel for this utterance.
- $SegAlign$: A list containing all the model alignments (two parameter values $T1$ and $T2$ for every diphone segment) of a single channel of an utterance.
- Utt_{model}: A series of parameter lists (one for every diphone transition). Three line pieces and two model alignment values are stored for each transition.
- Length: Function that returns the number of elements of a data structure.
- FirstOrderFit: Function that perform a least square error fit for a first-order polynomial function. A list of feature values and two scalar time alignment values is required as input. Then the function returns a single scalar number.
- DrawLine: Function that connects two Euclidean co-ordinates (in two-dimensional space), using a first-order line piece. Two lists, each containing the scalar co-ordinate input values, are required as input. Output is given as a list of polynomial coefficients, describing the first-order line.

Algorithm 5 Connect model

function CONNECTSEGMENTS(Seg_{feat},$SegAlign$)

  for $i = 1$ to $Length(Seg_{feat})$ **do** // ***Generate model on per-segment basis***
    $T1, T2 \leftarrow SegAlign_i$

    if $i = 1$ **then** // ***Additional stable value for first segment***
      $T3, T4 \leftarrow SegAlign_{i+1}$
      $S1 \leftarrow FirstOrderFit(Seg_{feat}(i), 1, T1)$
      $S2 \leftarrow FirstOrderFit(Seg_{feat}(i), T2, T3)$
      $Change \leftarrow DrawLine([T1, S1], [T2, S2])$
      $trans_{list} \leftarrow [S1, Change, S2, T1, T2]$
      Add $trans_{list}$ to Utt_{model}
      $S2_{prev} \leftarrow S2$ // ***Connect to next segment***

    else if $i = Length(Seg_{feat})$ **then** // ***Last segment***
      $Seg_{end} \leftarrow Length(Feat)$
      $S2 \leftarrow FirstOrderFit(Seg_{feat}(i), T2, Seg_{end})$
      $Change \leftarrow DrawLine([T1, S2_{prev}], [T2, S2])$
      $trans_{list} \leftarrow [S2_{prev}, Change, S2, T1, T2]$
      Add $trans_{list}$ to Utt_{model}

    else // ***Only 2 line pieces for all other segments***
      $T3, T4 \leftarrow SegAlign_{i+1}$
      $S2 \leftarrow FirstOrderFit(Seg_{feat}(i), T2, T3)$
      $Change \leftarrow DrawLine([T1, S2_{prev}], [T2, S2])$
      $trans_{list} \leftarrow [S2_{prev}, Change, S2, T1, T2]$
      Add $trans_{list}$ to Utt_{model}
      $S2_{prev} \leftarrow S2$ // ***Connect to next segment***
    end if
  end for
return Utt_{model}
end function

B.2 Predicting trajectories

References must first be obtained to predict a set of trajectories using reference stable values, as described in Section 7.3.2. The algorithm used for the estimation of such a set of values appears below (Section B.2.1). Furthermore, applying reference values as stable value predictors and constructing a complete set of test trajectories can be accomplished by using Algorithm 7, shown in Section B.2.2.

B.2.1 Reference stable values

The description of the variables and functions required by the algorithm to estimate stable value predictors (Algorithm 6):

- Utt_{Models}: A list of Utt_{model} data structures (one for every utterance).
- $Seg_{contexts}$: Data structure with context labels for each diphone segment of every utterance.
- $Stable_{list}$: Data structure containing all stable values for particular groups of contexts. When context labels are not symmetrical (such as biphone of 4-phone context sizes), the context label of the same stable value (part of two segments, connecting the two) will differ in both the segments where it occurs. For this reason, the algorithm adds two stable values of every segment, even though it produces two examples of the same value for symmetrical context labels.
- $Stable_{ref}$: A data structure containing all the stable reference values (for every context).
- GetContext: Function that generates the stable value context labels of stable values $S1_{context}$ and $S2_{context}$. Given an index for utterance as well as diphone, a context for the required diphone is selected.
- Length: Function that returns the number of elements of a data structure.
- Mean: Function that returns the mean value of a list of scalar values.

Algorithm 6 Reference stable values

```
1: function CalculateStableReferences(Utt_Models, Seg_contexts)
2:
3:     for u = 1 to Length(Utt_Models) do  //  for every utterance
4:         for i = 1 to Length(Utt_Models(u)) do  //  for every segment
5:             (S1_context, S2_context) ← GetContext(Seg_contexts, u, i)
6:             S1 ← Utt_Models[u][i]['S1']
7:             add S1 to Stable_list(S1_context)
8:             S2 ← Utt_Models[u][i]['S2']
9:             add S2 to Stable_list(S2_context)
10:        end for
11:    end for
12:
13:    for all context in Stable_list do
14:        Stable_ref(context) ← Mean(Stable_list(context))
15:    end for
16: return Stable_ref
17: end function
```

B.2.2 Predicted test features

Given the feature values of all diphone segments of an utterance, Algorithm 7 finds the model alignments ($T1$ and $T2$) for every segment and fits reference stable values to predict the test trajectory of a single channel. The model alignments can be obtained by using Algorithm 1. In the later work (see Section 7.8), model alignments were generated by using Algorithms 1 and 4, but instead of fitting stable values directly to the test features, the selected reference stable value (taken from $Stable_{ref}$ below) was used as the stable value options in the alignment algorithms. The best fit for the predicted test trajectory was found.

The variables and functions used for predicting a set of test trajectories (Algorithm 7) are described as follows:

- Seg_{Feat}: All the diphone segments of a single feature channel for this utterance.
- $Stable_{ref}$: A data structure containing all stable reference values (one for every context).
- $Seg_{contexts}$: Data structure with context labels for each diphone segment of every utterance. Context labels are stored for both stable values, $S1_{context}$ and $S2_{context}$ contexts.
- u: The index of the current utterance.
- GetAlign: Function that selects and executes the algorithm used for generating the model alignments.
- Mean: Function that returns the mean value of a list of scalar values.
- Length: Function that returns the number of elements of a data structure.

Algorithm 7 Predict the trajectory of unseen test data

function PREDICTEDUTT(Seg_{Feat}, $Stable_{ref}$, $Seg_{contexts}$, u)

  $SegAlign \leftarrow GetAlign(Seg_{feat}, Stable_{ref}, Seg_{contexts})$ // ***Obtain model alignments for utterance***

  for $i = 1$ to $Length(Seg_{feat})$ **do** // ***Generate model on per-segment basis***
    $(S1_{context}, S2_{context}) \leftarrow Seg_{contexts}[u][i]$
    $(S3_{context}, S4_{context}) \leftarrow Seg_{contexts}[u][i+1]$
    $T1, T2 \leftarrow SegAlign_i$
    $T3, T4 \leftarrow SegAlign_{i+1}$

    if i = 1 **then** // ***First segment***
      $S1 \leftarrow Stable_{ref}[S1_{context}]$
      $S2 \leftarrow Mean(Stable_{ref}[S2_{context}], Stable_{ref}[S3_{context}])$
      $Change \leftarrow DrawLine([T1, S1], [T2, S2])$
      $trans_{list} \leftarrow [S1, Change, S2, T1, T2]$
    else if $i = Length(Seg_{feat})$ **then** // ***Last segment***
      $S2 \leftarrow Stable_{ref}[S2_{context}]$
      $Change \leftarrow DrawLine([T1, S2_{prev}], [T2, S2])$
      $trans_{list} \leftarrow [S2_{prev}, Change, S2, T1, T2]$
    else // ***Two lines (S2 and Change) for all other segments***
      $S2 \leftarrow Mean(Stable_{ref}[S2_{context}], Stable_{ref}[S3_{context}])$
      $Change \leftarrow DrawLine([T1, S2_{prev}], [T2, S2])$
      $trans_{list} \leftarrow [S2_{prev}, Change, S2, T1, T2]$
    end if

    Add $trans_{list}$ to Utt_{model}
    $S2_{prev} \leftarrow S2$
  end for
return Utt_{model}
end function

B.3 Improving the stable values on turning points

The position of the stable values that occur at local minima or maxima in the feature channel of a whole utterance were optimized to obtain a better model fit. The "RefitStable" algorithm (Algorithm 9 below) can be used together with the "Connect-Segments" algorithm (Algorithm 5) to improve the stable values of an utterance model (Utt_{model}). From such a model, the current and previous diphone segments and the corresponding feature values are re-evaluated to search for the best stable value (connecting the two segments). The "RefitStable" algorithm calculates the model fit of all possible alignments for a new stable value, using the "StableFit" Algorithm 8. First, the variables and functions required by the "StableFit"algorithm are described below:

- $Feat$: The feature values (one for each frame) for a single channel.

Algorithm 8 Calculate MSE for stable value fit option

```
1: function StableFit(Feat, T1_C1, T2_C1, T1_C2, T2_C2, Prev_S, New_S, Next_S)
2:
3:     SE_total ← 0
4:     Prev_C ← DrawLine([T1_C1, Prev_S], [T2_C1, New_S])
5:     if T2_C1 > T1_C1 + 1 then
6:         SE_total ← SE_total + SE(Prev_C, Feat, T1_C1 + 1, T2_C1 − 1)
7:     end if
8:     SE_total ← SE_total + SE(New_S, Feat, T2_C1, T1_C2)
9:     Next_C ← DrawLine([T1_C2, New_S], [T2_C2, Next_S])
10:    if T2_C2 > T1_C2 + 1 then
11:        SE_total ← SE_total + SE(Next_C, Feat, T1_C2 + 1, T2_C2 − 1)
12:    end if
13:    MSE ← SE_total/(T2_C2 − T1_C1)
14:    MSE_option ← [MSE, T1_C1, T2_C1, Prev_C, T1_C2, T2_C2, Next_C]
15: return MSE_option
16: end function
17:
```

- $T1_{C1}$: The $T1$ alignment index (the change descriptor start position) of the first segment.
- $T2_{C1}$: The $T2$ alignment index (the change descriptor end position) of the first segment.
- $T1_{C2}$: The $T1$ alignment index (the change descriptor start position) of the second segment.
- $T2_{C2}$: The $T2$ alignment index (the change descriptor end position) of the second segment.
- $Prev_S$: First stable value of the first segment.
- New_S: Second stable value of the first segment connecting the first and second segments.
- $Next_S$: Second stable value of the second segment.
- SE: Function computing the squared error. Similarly to the FirstOrderFit function, two time-alignment values and a list of feature values are needed. In addition, a polynomial line piece (list of coefficients) is required as the feature estimator for which the error is then computed. Then the squared error itself is returned as a single scalar value.

- DrawLine: Function that connects two Euclidean co-ordinates (in two-dimensional space), using a first-order line piece. Two lists, each containing the scalar co-ordinate input values, are required as input. Output is given as a list of polynomial coefficients, describing the first order line.

Algorithm 9 Improve stable value fit

```
1: function RefitStable(Utt_model, Seg_feat, i)
2:
3:     Change_cur ← Slope(Utt_model(i)) //  Detect local maximum/minimum
4:     Change_prev ← Slope(Utt_model(i − 1))
5:     if OppositeSign(Change_cur, Change_prev)) then
6:         Fit_parm ← GetParams(Utt_model[i − 1..i])
7:         Curr_fit ← StableFits(Feat, Fit_parm)
8:         MSE_fits ← AddOption(MSE_fits, Curr_fit) //  Include current fit
9:         New_S ← MaxValue(Feat, Fit_parm) //  Choose local optimum
10:
11:        for T2_C1 = T1_C1 + 1 to T2_C2 − 3 do
12:            for T1_C2 = T2_C1 + 2 to T2_C2 − 1 do  //  Min stable value length: 1
13:                Add T1_C2, T2_C1 and New_S to Fit_parms
14:                New_fit ← StableFit(Feat, Fit_parm)
15:                MSE_fits ← AddOption(MSE_fits, New_fit)
16:            end for
17:        end for
18:        Best_fit ← BestOption(MSE_fits) //  Find best alignments
19:        Update Utt_model with new Best_fit
20:    end if
21: return Utt_model
22: end function
```

Next the variables and functions required by the "RefitStable" algorithm are as follows:

- Utt_{model}: A series of parameter lists (one for every diphone transition). Three line pieces and two model alignment values are stored for each transition.
- Seg_{Feat}: All the diphone segments of a single feature channel for this utterance.
- i: The segment index of the current segment.
- Slope: Function that looks up the slope of a change descriptor given the model parameters of particular segment.
- OppositeSign: Function to determine whether or not the change descriptors of the first and second segment have slope values of the same sign.
- GetParams: Function combining all the required model parameters of the two segments into one array.

- AddOption: Function to add the model parameters of an alignment option to a list.
- MaxValue: Function to select the feature value with the absolute maximum magnitude from a set of feature values.
- BestOption: Function to select the series of model parameters providing the lowest error.

References

[1] J. S. Garofolo, Lori F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren. The DARPA TIMIT acoustic-phonetic continuous speech corpus, NIST order number PB91-100354, February 1993.

[2] K-F. Lee. *Large-vocabulary speaker-independent continuous speech recognition: The Sphinx system*. PhD thesis, Carnegie Mellon University, 1988.

[3] L. Deng, G. Ramsay, and D. Sun. Production models as a structural basis for automatic speech recognition. *Speech Communication*, 33(2-3):93–111, April 1997.

[4] D. Yu, L. Deng, and A. Acero. A lattice search technique for a long-contextual-span hidden trajectory model of speech. *Speech Communication*, 48(9):1214–1226, September 2006.

[5] V. Digalakis. *Segment-Based Stochastic Models of Spectral Dynamics for Continuous Speech Recognition*. PhD thesis, Boston University, 1992.

[6] W.J. Holmes and M. J. Russell. Probabilistic-trajectory segmental HMMs. *Computer Speech and Language*, 13(1):3–37, January 1999. URL http://www.idealibrary.com.

[7] I-F. Chen and H-M. Wang. Articulatory feature asynchrony analysis and compensation in detection-based ASR. In *Proceedings of Interspeech*, pages 3059–3062, Brighton, United Kingdom, September 2009.

[8] G. Zweig and P. Nguyen. SCARF: A segmental conditional random field toolkit for speech recognition. In *Proceedings of Interspeech*, pages 2858–2861, Makuhari, Chiba, Japan, September 2010.

[9] G. Zweig, P. Nguyen, D. van Compernolle, K. Demuynck, L. Atlas, P. Clark, G. Sell, M. Wang, F. Sha, H. Hermansky, D. Karakos, A. Jansen, S. Thomas, G.S.V.S. Sivaram, S. Bowman, and J. Kao. Speech recognition with segmental conditional random fields: A summary of the JHU CLSP 2010 summer workshop. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference*, pages 5044–5047, Prague, Czech Republic, May 2011.

[10] L. Besacier, V-B. Le, C. Boitet, and V. Berment. ASR and translation for under-resourced languages. In *Acoustics, Speech, and Signal Processing (ICASSP), 2006 IEEE International Conference on*, Toulouse.

[11] M.P. Harper. The Babel research program, 2015. http://www.iarpa.gov/index.php/research-programs/babel.

[12] J. Mamou, J. Cui, X. Cui, M.J.F. Gales, B. Kingsbury, K. Knill, L. Mangu, D. Nolden, M. Picheny, B. Ramabhadran, et al. Developing keyword search under the IARPA Babel program. In *Proceedings of Afeka Speech Processing Conference*, Tel Aviv, July 2013.

[13] H. Chang and J. Glass. Multi-level context-dependent acoustic modeling for automatic speech recognition. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 89–94, Waikoloa, HI, December 2011.

[14] S. J. Young, J. J. Odell, and P. C. Woodland. Tree-based state tying for high accuracy acoustic modelling. In *Proceedings of ARPA Workshop on Human Language Technology*, pages 307–312, Plainsboro, March 1994.

[15] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A.W. Black, and K. Tokuda. The HMM-based speech synthesis system (HTS) version 2.0. In *Proceedings of the 6th ISCA Workshop on Speech Synthesis*, pages 294–299, Bonn, Germany, August 2007.

[16] X. Lu, S. Matsuda, M. Unoki, and S. Nakamura. Temporal contrast normalization and edge-preserved smoothing of temporal modulation structures of speech for robust speech recognition. *Speech Communication*, 52(1):1–11, January 2010.

[17] J. Dines, J. Yamagishi, and S. King. Measuring the gap between HMM-based ASR and TTS. *IEEE Journal of Selected Topics in Signal Processing*, 4(6):1046 – 1058, December 2010.

[18] S. Furui. Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 29(2):254–272, April 1981.

[19] H. Zen, K. Tokuda, and A.W. Black. Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039–1064, April 2009.

[20] J.A. Bilmes. Buried Markov models for speech recognition. In *Proceedings of Acoustics, Speech, and Signal Processing, 1999 IEEE International Conference*, volume 2, pages 713–716, Phoenix, Arizona, USA, March 1999.

[21] M. Gales and S. Young. The application of hidden Markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304, January 2008.

[22] A-V.I. Rosti and M.J.F. Gales. Factor analysed hidden Markov models for speech recognition. *Computer Speech and Language*, 18(2):181–200, September 2004.

[23] D. Povey, L. Burget, M. Agarwal, P. Akyazi, F. Kai, A. Ghoshal, O. Glembek, N. Goel, M. Karafiát, A. Rastrow, et al. The subspace gaussian mixture model – A structured model for speech recognition. *Computer Speech & Language*, 25(2): 404–439, April 2011.

[24] G.E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, April 2012.

[25] A. Biem, S. Katagiri, E. McDermott, and B.H. Juang. An application of discriminative feature extraction to filter-bank-based speech recognition. *IEEE Transactions on Speech and Audio Processing*, 9(2):96–110, February 2001.

[26] O. Viikki and K. Laurila. Cepstral domain segmental feature vector normalization for noise robust speech recognition. *Speech Communication*, 25(1-3):133147, August 1998.

[27] O.M. Strand and A. Egeberg. Cepstral mean and variance normalization in the model domain. In *In COST278 and ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction.*, University of East Anglia, Norwich, UK, August 2004.

[28] L. Lee and R.C. Rose. Speaker normalization using efficient frequency warping procedures. In *Proceedings of Acoustics, Speech, and Signal Processing, 1996 IEEE International Conference*, volume 1, pages 353–356, Atlanta, GA, May 1996.

[29] T. Hain, P.C. Woodland, T.R. Niesler, and E.W.D. Whittaker. The 1998 HTK system for transcription of conversational telephone speech. In *Proceedings of Acoustics, Speech, and Signal Processing, 1999 IEEE International Conference*, volume 1, pages 57–60, Phoenix, AZ, March 1999.

[30] D.Y. Kim, S. Umesh, M.J.F. Gales, T. Hain, and P.C. Woodland. Using VTLN for broadcast news transcription. In *Proceedings of Interspeech*, Jeju Island, Korea, October 2004.

[31] D.R. Sanand and S. Umesh. VTLN using analytically determined linear-transformation on conventional MFCC. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(5):1573–1584, July 2012.

[32] S. Panchapagesan and A. Alwan. Frequency warping for VTLN and speaker adaptation by linear transformation of standard MFCC. *Computer speech and language*, 23(1):42–64, March 2008.

[33] M. Afify, X. Cui, and Y. Gao. Stereo-based stochastic mapping for robust speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 17(7): 1325–1334, September 2009.

[34] H. Zen, Y. Nankaku, and K. Tokuda. Continuous stochastic feature mapping based on trajectory HMMs. *IEEE Transactions on Audio, Speech and Language Processing*, 19(2):417–430, May 2010.

[35] H. Zen, K. Tokuda, and T. Kitamura. Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences. *Computer Speech and Language*, 21(1):153173, January 2007.

[36] C. P. Chen and J. A. Bilmes. MVA processing of speech features. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):257–270, January 2007.

[37] X. Xiao, S. Chng, and H. Li. Normalization of the speech modulation spectra for robust speech recognition. *IEEE Transactions on Computer Speech and Language*, 16(8):1662–1674, November 2008.

[38] O. Ghitza. Auditory models and human performance in tasks related to speech coding and speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):115–132, January 1994.

[39] J.W. Pitton, K. Wang, and B.H. Juang. Time-frequency analysis and auditory modeling for automatic recognition of speech. *Proceedings of the IEEE*, 84(9): 1199–1215, September 1996.

[40] D. Dimitriadis, P. Maragos, and A. Potamianos. On the effects of filterbank design and energy computation on robust speech recognition. *IEEE Transactions on Computer Speech and Language*, 19(6):1504–1516, August 2011.

[41] S. D. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, August 1980.

[42] M. J. Gales. Semi-tied covariance matrices for hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 7(3):272–281, May 1999.

[43] N. Malayath and H. Hermansky. Data-driven spectral basis functions for automatic speech recognition. *Speech Communication*, 4(4):449–466, June 2003.

[44] R. Haeb-Umbach and H. Ney. Linear discriminant analysis for improved large vocabulary continuous speech recognition. In *Proceedings of Acoustics, Speech, and Signal Processing, 1992 IEEE International Conference*, volume 1, pages 13–16, San Francisco, CA, March 1992.

[45] A.M. Martínez and A.C. Kak. PCA versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):228–233, February 2001.

[46] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen. Maximum likelihood discriminant feature spaces. In *Proceedings of Acoustics, Speech, and Signal Processing, 2000 IEEE International Conference*, volume 2, pages II1129–II1132, Istanbul, June 2000.

[47] N. Kumar and A.G. Andreou. Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition. *Speech communication*, 26(4):283–297, December 1998.

[48] L.A. Liporace. Maximum likelihood estimation for multivariate observations of markov sources. *IEEE Transactions on Information Theory*, 28(5):5, September 1982.

[49] R.A. Gopinath. Maximum likelihood modeling with gaussian distributions for classification. In *Proceedings of Acoustics, Speech, and Signal Processing, 1998 IEEE International Conference*, volume 2, pages 661–664, Seattle, WA, May 1998.

[50] J. Gauvain and C. Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298, April 1994.

[51] M. Benzeghiba, R. De Mori, O. Deroo, S. Dupont, T. Erbes, D. Jouvet, L. Fissore, P. Laface, A. Mertins, C. Ris, et al. Automatic speech recognition and speech variability: A review. *Speech Communication*, 49(10):763–786, February 2007.

[52] M.J.F. Gales and P.C. Woodland. Mean and variance adaptation within the MLLR framework. *Computer Speech and Language*, 10(4):249264, October 1996.

[53] S.M. Ahadi and P.C. Woodland. Combined Bayesian and predictive techniques for rapid speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*, 11(3):187206, July 1997.

[54] C.J. Leggetter and P.C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*, 9(2):171–185, April 1995.

[55] P.C. Woodland, D. Pye, and M.J.F. Gales. Iterative unsupervised adaptation using maximum likelihood linear regression. In *Spoken Language (ICSLP), 1996 IEEE International Conference*, volume 2, pages 1133–1136, October 1996.

[56] M.J.F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, 12(2):7598, April 1998.

[57] O. Siohan, T.A. Myrvoll, and C.H. Lee. Structural maximum a posteriori linear regression for fast HMM adaptation. *Computer Speech and Language*, 16(1):5–24, January 2002.

[58] Y. Nakano, M. Tachibana, J. Yamagishi, and T. Kobayashi. Constrained structural maximum a posteriori linear regression for average-voice-based speech synthesis. In *Proceedings of Interspeech*, page 22862289, Pittsburgh, Pennsylvania, USA, September 2006.

[59] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul. A compact model for speaker-adaptive training. In *Proceedings of Interspeech*, pages 1137–1140, Philadelphia, PA, October 1996.

[60] P.C. Woodland and D. Povey. Large scale discriminative training of hidden Markov models for speech recognition. *Computer Speech and Language*, 16(1):25–47, January 2002.

[61] R. Schlüter, W. Macherey, B. Müller, and H. Ney. Comparison of discriminative training criteria and optimization methods for speech recognition. *Speech Communication*, 34(3):287–310, June 2001.

[62] A. Nádas. A decision theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 31(4): 814–817, August 1983.

[63] B-H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Transactions on Signal Processing*, 40(12):3043–3054, December 1992.

[64] D. Povey and P.C. Woodland. Minimum phone error and i-smoothing for improved discriminative training. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, pages I–105–I–108, Orlando, FL, May 2002.

[65] J. Zheng and A. Stolcke. Improved discriminative training using phone lattices. In *Proceedings of Interspeech*, pages 2125–2128, Lisbon, Portugal, September 2005.

[66] V. Valtchev, J.J. Odell, P.C. Woodland, and S.J. Young. MMIE training of large vocabulary recognition systems. *Speech Communication*, 22(4):303–314, September 1997.

[67] A. Mohamed, T.N. Sainath, G. Dahl, B. Ramabhadran, G.E. Hinton, M. Picheny, et al. Deep belief networks using discriminative features for phone recognition.

In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference*, pages 5060–5063, Prague, May 2011.

[68] A. Mohamed, G. Hinton, and G. Penn. Understanding how deep belief networks perform acoustic modelling. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference*, pages 4273–4276, Kyoto, March 2012.

[69] Y. Huang, D. Yu, C. Liu, and Y. Gong. A comparative analytic study on the gaussian mixture and context dependent deep neural network hidden Markov models. In *Proceedings of Interspeech*, pages 1895–1899, Singapore, September 2014.

[70] T.N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A. Mohamed. Making deep belief networks effective for large vocabulary continuous speech recognition. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop*, pages 30–35, Waikoloa, HI, December 2011.

[71] K.C. Sim and M.J.F. Gales. Discriminative semi-parametric trajectory model for speech recognition. *Computer Speech and Language*, 21(4):669–687, October 2007.

[72] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, and G. Zweig. fMPE: Discriminatively trained features for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2005 IEEE International Conference*, pages 961–964, Philadelphia, PA, March 2005.

[73] K.C. Sim and M.J.F Gales. Temporally varying model parameters for large vocabulary continuous speech recognition. In *Proceedings of Interspeech*, pages 2137–2140, Lisbon, Portugal, September 2005.

[74] K. Tokuda, H. Zen, and T. Kitamura. Trajectory modeling based on HMMs with the explicit relationship between stochastic and dynamic features. In *Proceedings of Eurospeech*, pages 865–868, Geneva, Switzerland, September 2003.

[75] M. Ostendorf, V. Digalakis, and O. Kimball. From HMMs to segment models: A unified view of stochastic speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 4(5):360–378, May 1996.

[76] A-V. I. Rosti and M.J.F. Gales. Switching linear dynamical systems for speech recognition. In *Technical Report CUED/F-INFENG/TR461*, Cambridge University, 2003.

[77] A. Ragni, K.M. Knill, S.P. Rath, and M.J.F. Gales. Data augmentation for low resource languages. In *Proceedings of Interspeech*, pages 810–814, Singapore, September 2014.

[78] M.J.F. Gales, D.Y. Kim, P.C. Woodland, H.Y. Chan, D. Mrva, R. Sinha, and S.E. Tranter. Progress in the CU-HTK broadcast news transcription system. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1513–1525, September 2006.

[79] Y. Qian, K. Yu, and J. Liu. Combination of data borrowing strategies for low-resource LVCSR. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 404–409, Olomouc, Czech Republic, December 2013. IEEE.

[80] H. Lin, L. Deng, D. Yu, Y. Gong, A. Acero, and C-H. Lee. A study on multilingual acoustic modeling for large vocabulary ASR. In *Acoustics, Speech and Signal Processing (ICASSP), 2009 IEEE International Conference on*, pages 4333–4336, Taipei, April 2009.

[81] N. Jaitly and G.E. Hinton. Vocal tract length perturbation (VTLP) improves speech recognition. In *Proceedings of ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[82] X. Cui, V. Goel, and B. Kingsbury. Data augmentation for deep neural network acoustic modeling. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference*, pages 5582–5586. IEEE, 2014.

[83] N. Kanda, R. Takeda, and Y. Obuchi. Elastic spectral distortion for low resource speech recognition with deep neural networks. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 309–314, Olomouc, Czech Republic, December 2013. IEEE.

[84] M.J.F. Gales, A. Ragni, H. AlDamarki, and C. Gautier. Support vector machines for noise robust ASR. In *Automatic Speech Recognition and Understanding (ASRU), 2009 IEEE Workshop on*, pages 205–210, Merano, Italy, November 2009. IEEE.

[85] The CMU pronunciation dictionary, 2010. http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

[86] N. J. de Vries, J. Badenhorst, M.H. Davel, E. Barnard, and A. de Waal. Woefzela – an open-source platform for ASR data collection in the developing world. In *Proceedings of Interspeech*, pages 3177–3180, Florence, Italy, August 2011.

[87] E. Barnard, M.H. Davel, C. van Heerden, F. de Wet, and J. Badenhorst. The NCHLT speech corpus of the South African languages. In *Proceedings of spoken languages technologies for under-resourced languages*, pages 194–200, St Petersburg, Russia, May 2014.

[88] D. Gibbon, R. Moore, and R. Winski. *Handbook of standards and resources for spoken language systems*. Walter de Gruyter, 1997.

[89] N.J. de Vries, M.H. Davel, J. Badenhorst, W.D. Basson, F. de Wet, E. Barnard, and A. de Waal. A smartphone-based ASR data collection tool for under-resourced languages. *Speech Communication*, 56(0):119–131, January .

[90] C. van Heerden M. Davel and E. Barnard. Validating smartphone-collected speech corpora. In *Proceedings of spoken languages technologies for under-resourced languages*, pages 68–75, Cape Town, South Africa, May 2012.

[91] M.H. Davel, C. van Heerden, N. Kleynhans, and E. Barnard. Efficient harvesting of internet audio for resource-scarce ASR. In *Proceedings of Interspeech*, pages 3153–3156, Florence, Italy, August 2011.

[92] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Veltchev, and P. Woodland. *The HTK Book*. Cambridge University Engineering Department, http://htk.eng.cam.ac.uk/, 2009.

[93] D. Wissing. Derounding in Afrikaans, 2011. http://www.scielo.org.za/pdf/tvg/v51n1/v51n1a01.pdf.

[94] J. Badenhorst, M.H. Davel, and E. Barnard. Trajectory behaviour at different phonemic context sizes. In *Proceedings of PRASA*, pages 1–6, Vanderbijlpark, South Africa, November 2011.

[95] M. Davel and E. Barnard. Pronunciation prediction with Default&Refine. *Computer Speech and Language*, 22(4):374393, October 2008.

[96] H. Davel and F. de Wet. Verifying pronunciation dictionaries using conflict analysis. In *Proceedings of Interspeech*, pages 1898–1901, Makuhari, Chiba, Japan, September 2010.

[97] J. A. C. Badenhorst, M. H. Davel, and E. Barnard. Analysing co-articulation using frame-based feature trajectories. In *Proceedings of PRASA*, pages 13–18, Stellenbosch, South Africa, November 2010.

[98] J. Badenhorst, M.H. Davel, and E. Barnard. Improved transition models for cepstral trajectories. In *Proceedings of PRASA*, pages 157–164, Pretoria, South Africa, November 2012.

[99] J. Badenhorst and M.H. Davel. Synthetic triphones from trajectory-based feature distributions. In *Accepted for publication in proceedings of PRASA*, Port Elizabeth, South Africa, November 2015.