The Effect of Imperfect Cues on the Reception of Cued Speech

by

Maroula Sarah Bratakos

S.B., Massachusetts Institute of Technology (1993)

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1995

© Massachusetts Institute of Technology 1995. All rights reserved.

Author

Department of Electrical Engineering and Computer Science

August 23, 1995

Certified by

Louis D. Braida

Henry E. Warren Professor of Electrical Engineering Thesis Supervisor

Accepted by

Frederic R. Morgenthaler

Chairman, Department Committee on Graduate Theses

JAN 29 1996
The Effect of Imperfect Cues on the Reception of Cued Speech

by

Maroula Sarah Bratakos

Submitted to the Department of Electrical Engineering and Computer Science on August 23, 1995, in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering

Abstract

Manual Cued Speech (MCS) is a supplement to speechreading that reduces lipreading errors by reducing the ambiguity of lip shapes. The effects of cues on the reception of Cued Speech were studied to determine the feasibility of building a real-time automatic cueing system that would employ a phonetic speech recognizer. Synthetic cues based on the groups of MCS were superimposed on prerecorded sentences and simulated phone errors consisting of substitutions, deletions and insertions were introduced at rates similar to those of current ASRs. The effect of delayed cue presentation was also explored. The synthetic cues were discrete and presented at a slightly faster rate than MCS. Five deaf, experienced cue receivers were tested on their ability to speechread key words in sentences under 12 synthetic cue conditions. Results were compared with their scores on SA and MCS. Although scores on the perfect synthetic cues (PSC) condition did not quite attain the high scores obtained with MCS, differences between the two conditions were not statistically significant. Subjects seemed unaffected by the discrete nature of the cues and the faster presentation. Average scores for conditions with 10% and 20% phone error rates were 69% and 60% correct, respectively. The average score on the REAL condition, which contained sentences processed by an actual phonetic recognizer and a 21% phone error rate, was 70% correct. Subjects performed better on REAL because its key word error rate was low, probably due to phone errors occurring nonuniformly throughout the sentences. In contrast, the phone errors were uniformly dispersed in the 10% and 20% error conditions. Results on other conditions suggest that deletions affect subjects' abilities to speechread sentences more than insertions. In addition, cue presentation delays of more than 33msec greatly reduce speechreading scores. On all 12 synthetic cue conditions subjects achieved higher scores than with SA indicating that even erroneous cues would aid those who speechread. The results of this study, in conjunction with recent advances in phonetic speech recognizers, suggest that building a real-time automatic cueing system is feasible and would be beneficial to deaf cue receivers.

Thesis Supervisor: Louis D. Braida
Title: Henry E. Warren Professor of Electrical Engineering
Acknowledgments

There are many people I want to thank for their help and support over the past year, without whom I would not have made it so far so fast (yes, these are the sappy thank yous that Academy Award speeches are made of). First and foremost, I have to thank my advisor, Lou Braida, who was always there with suggestions, advice and a helping hand (not to mention a camera at my water polo games). His optimism and support throughout the year were invaluable.

Of course, this project would not have been possible without Paul Duchnowski. He provided the ASR errors and expertise that went into my thesis, along with the wisdom on thesis research to keep me sane. Thanks also to Fred and Becky who did the little tasks that were amazingly boring but extremely important. Speaking of boring, thanks to Lorraine Delhorne who helped me record (and rerecord) hundreds of sentences and test subjects on the weekends. It’s been fun exchanging stories with her and her cohort Charlotte Reed. Charlotte got me hooked on the seventh floor when I did my senior thesis with her, so a special thanks goes out to her.

Thanks to the ‘boys’ office’ (Joe, Phil, and Dave) for letting me hang out in their doorway and helping me with my subjects. Also thanks to all the people not in the lab who offered friendship and support: Clara, Cheryl, Tom, the women’s water polo team and especially my mom, John and Tracy.

The most important group of people I need to thank are my subjects. They are amazing individuals who have achieved so much despite being deaf. I have a great respect for them and their accomplishments. I especially want to thank Sarah, who more than 15 years ago introduced me to the world of deafness and Cued Speech. She provided the motivation for this thesis. Also, thank you to my other motivator, Stasie, who had the patience to teach me how to Cue, always had a smile and encouraging word, and was a great friend.

Last, but not least, thank you to the best officemate ever, Jeanie. Without ‘the beach’ I don’t think either of us would have made it. Jeanie, I’m going to miss you. Thanks for being a friend.
## Contents

1 Introduction ......................................................... 9
   1.1 Background ................................................. 9
   1.2 Objective .................................................. 12

2 Methods ...................................................................... 13
   2.1 Speech Materials ............................................ 13
   2.1.1 CID Sentences .......................................... 14
   2.1.2 Clarke Sentences ....................................... 14
   2.1.3 Harvard Sentences ..................................... 14
   2.2 Recording of Synthetic Cues .............................. 15
   2.2.1 The Cues .................................................. 15
   2.2.2 The Playlists ............................................. 15
   2.2.3 Converting Times to Frames ......................... 16
   2.2.4 Converting Phones to Cues ......................... 16
   2.2.5 Recording ................................................. 17
   2.3 Subjects ........................................................ 18
   2.4 Experimental Conditions .................................... 19
   2.4.1 Phase I .................................................... 20
   2.4.2 Phase II .................................................... 22
   2.5 Procedure ...................................................... 25
   2.5.1 Phase I .................................................... 26
   2.5.2 Phase II .................................................... 27
3 Results

3.1 Data Analysis ........................................ 28
  3.1.1 Scoring .......................................... 28
  3.1.2 Statistical Analysis ................................. 28

3.2 Phase I .................................................. 29

3.3 Phase II .................................................. 31

3.4 Comparisons across Phases ............................... 34

3.5 Phone Error Rates vs Key Word Error Rates .............. 36

3.6 Comparison to Autocuer Study .......................... 40

4 Discussion .................................................. 43

4.1 Effect of Synthetic Cues ................................ 43

4.2 Effect of Error Rate .................................... 45

4.3 Effect of Delay .......................................... 49

4.4 Choice of Cue Groups ................................... 49

4.5 Summary .................................................. 50

A Auxiliary Experiment: The Effect of Limited Hand Movements on the Reception of Cued Speech 55

A.1 Purpose .................................................. 55

A.2 Methods .................................................. 55

A.3 Results ................................................... 56

A.4 Discussion ............................................... 56

B Statistical Analysis ....................................... 58

B.1 ANOVA .................................................... 58

B.2 F Test ...................................................... 60

B.3 t-tests ...................................................... 61

C Raw Data .................................................... 63

D Transcription Files and Playlists ........................... 67
List of Figures

2-1 Block diagram of system used to superimpose cues. ............... 18

3-1 Error conditions of Phase I (no delay). ......................... 30
3-2 Delay conditions of Phase I. .............................. 31
3-3 Random delay conditions of Phase II. ....................... 33
3-4 Conditions containing 20% errors in Phase II. ............... 34
3-5 REAL condition compared with INS and RD2. ............... 35
3-6 Comparison of error conditions across phases. ............... 36
3-7 Comparison of delay conditions across phases. ............... 37
3-8 Phase I. .............................................. 40
3-9 Phase II. .............................................. 41
List of Tables

1.1 Cue Groups of Manual Cued Speech. .......................... 10

2.1 Histories of subjects experienced in receiving Manual Cued Speech (MCS). ........................................... 18
2.2 Experimental Conditions for Phase I. .......................... 21
2.3 Experimental Conditions for Phase II. .......................... 23

3.1 Average test scores for subjects in Phase I. ...................... 29
3.2 Average test scores for subjects in Phase II. ...................... 32
3.3 Conditions SA, MCS and PSC across both Phases. .............. 35
3.4 Phone Error Rate and Key Word Error Rate. .................... 38
3.5 Percent of key words containing errors correctly identified. ...... 39
3.6 Percent of key words containing no errors correctly identified. ... 39

A.1 Scores for Limited Hand Movement Study ....................... 57
A.2 Harvard sentences (59–72) position coordinates for LM. ........ 57

B.1 Analysis of Variance for Condition-Subject Interaction ........... 59
B.2 F Tests. .................................................................. 60
B.3 t-tests. .................................................................. 62

C.1 Phase I Training Results. ........................................... 63
C.2 Phase I Testing Results. ............................................ 64
C.3 Phase II Training Results. .......................................... 65
C.4 Phase II Testing Results. ............................................ 66
D.1 CID sentences position coordinates ........................................ 73
D.2 Clarke sentences position coordinates .................................. 73
D.3 Harvard sentences (1–58) position coordinates ....................... 73
D.4 Harvard sentences (59–72) position coordinates ...................... 74
E.1 Confusion matrix CM1 used to generate errors in conditions SD1, SD2, 
   D1, D3, D5, RD1, RD2, MARK .................................................. 76
E.2 Confusion matrix CM2 used to generate errors in condition INS  .... 77
E.3 Number of input phones in the matrices CM1 and CM2 ............. 78
Chapter 1

Introduction

1.1 Background

Although other methods of communication are available to the deaf (e.g. Sign Language, fingerspelling), many rely on lipreading to understand spoken language, particularly when communicating with members of the hearing world. Unfortunately, many phonemes appear similar on the lips; some are even indistinguishable. Such confusions can result in miscomprehension of over fifty percent of the words in spoken conversations, even by the most experienced lipreaders. Phonemes that appear similar on the lips and are often confused with each other are sometimes grouped into categories called visemes, e.g. /p,b,m/, /f,v/, /w,r/, and /dh,th/ [1].

An alternative communication method for the hearing impaired, Manual Cued Speech, was developed to alleviate lipreading ambiguities by Dr. Orin Cornett in 1967. Eight consonant handshapes and five vowel hand positions are combined to provide real-time supplemental information about the syllables appearing on the lips [2]. Viseme group ambiguities are resolved by assigning a different cue to each phoneme in a viseme group. Thus, phonemes in a single cue group can be easily discriminated through vision, although the cues do not contain enough information in themselves to permit speech communication without lipreading [3]. The cue groups of Manual Cued Speech are listed in Table 1.1. Diphthongs are cued by sequentially combining two vowel groups. Uchanski et al. [4], and Nicholls and Ling [3] have conducted studies of
Table 1.1: Cue Groups of Manual Cued Speech.

<table>
<thead>
<tr>
<th>Consonant Handshapes</th>
<th>Phonemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Name (Number)</td>
<td>Phonemes</td>
</tr>
<tr>
<td>d-handshape (1)</td>
<td>/d,p,zh/</td>
</tr>
<tr>
<td>k-handshape (2)</td>
<td>/k,v,dh,z/</td>
</tr>
<tr>
<td>h-handshape (3)</td>
<td>/h,s,r/</td>
</tr>
<tr>
<td>n-handshape (4)</td>
<td>/n,b,wh/</td>
</tr>
<tr>
<td>t-handshape (5)</td>
<td>/t,m,f/</td>
</tr>
<tr>
<td>l-handshape (6)</td>
<td>/l,sh,w/</td>
</tr>
<tr>
<td>g-handshape (7)</td>
<td>/g,j,th/</td>
</tr>
<tr>
<td>ng-handshape (8)</td>
<td>/ng,y,th/</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vowel Positions</th>
<th>Phonemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Name</td>
<td>Phonemes</td>
</tr>
<tr>
<td>mouth</td>
<td>/ee,er/</td>
</tr>
<tr>
<td>chin</td>
<td>/aw,eh,oo/</td>
</tr>
<tr>
<td>throat</td>
<td>/ae,ih,u/</td>
</tr>
<tr>
<td>side-front</td>
<td>/ah,o/</td>
</tr>
<tr>
<td>side-down</td>
<td>/uh/</td>
</tr>
</tbody>
</table>

the benefits Cued Speech provide to experienced Cued Speech receivers. Both studies found a significant increase in speechreading scores on sentence materials when Cued Speech was employed. For example, Uchanski et al. tested four experienced Manual Cued Speech receivers on Clarke sentences and Harvard sentences\(^1\) under the conditions of speechreading alone (SA) and speechreading supplemented with Cued Speech (MCS). The average score for the Clarke sentences in the SA condition was 45% while in the MCS condition the average score was 97%. The average score for the more difficult Harvard sentences in the SA condition was 25% while the average for the MCS condition was 84% [4]. Thus, a hearing impaired person using Cued Speech can understand nearly all the words in an everyday conversation. Even in more challenging situations, where contextual cues are unavailable, the benefit that Cued Speech provides is enormous.

Unfortunately, only a small number of people know how to produce cues, which

\(^1\)The Clarke sentences cover everyday topics and employ contextual cues. By contrast, the Harvard sentences provide far fewer contextual cues.
limits the number of people who can communicate with deaf individuals via Cued Speech. Dr. Cornett and his associates at Gallaudet College and the Research Triangle Institute began development of the ‘Autocuer’, a device to produce cues automatically, in the early 70s. Their idea was to design a wearable, real-time system that was capable of analyzing speech and producing cues, similar to those of Cued Speech, and then presenting these cues via a seven-segment visual display [5]. The automatic speech recognizer (ASR) used in the most recent implementation of the Gallaudet-R.T.I. Autocuer had a phone recognition accuracy of 54%, with a substitution rate of 13% and a deletion rate of 33%. In 1981, six deaf students were tested using this prototype version of the Autocuer on isolated word identification. Rather than using handshapes, a matrix of light-emitting diodes was superimposed over a prerecorded videotape of the speakers. A fixed delay of 150msec was also introduced between onset of lip movement and cue presentation (to simulate processing delay). After 40 hours of training, the subjects were tested under seven conditions which simulated various error rates. The condition most comparable to the Autocuer’s ASR had a 59% accuracy with a 12% substitution rate and a 28% deletion rate. The average score on this condition was 67% compared with 59% uncued and 86% perfect cues [6]. The main problem with the Autocuer lies in its low phoneme identification accuracy and high deletion rate.

Two other conditions in the Autocuer study simulated more accurate ASRs. Both had phone recognition accuracies of 80%. The first condition simulated an 11% substitution rate and a 9% deletion rate. The average score for subjects on this condition was 77%. The second condition contained a 5.5% substitution rate and a 14.5% deletion rate with subjects scoring 82% on average [6].

Great strides have been made in the speech recognition field since the conception of the Autocuer. Although many ASRs are aimed at word recognition, several phonetic recognition systems have been developed. Lee and Hon have reported phone

---

2The errors made by ASRs can be categorized into three types: substitution (incorrect phone identification), deletion (no identification, although phone is present), and insertion (identification although phone is not present).
recognition accuracy of 66% on a speaker independent system using a right-context-dependent phone model. The substitution rate for this system was 19%, the deletion rate was 6% and the insertion rate was 7% [7]. Robinson and Fallside have achieved 69% phone recognition accuracy on a speaker-independent system using connectionist techniques [8]. Zissman achieved 67% phone recognition accuracy on a HTK speaker-independent system [9], with a substitution rate of 20%, a deletion rate of 6% and an insertion rate of 7% [10]. On a HTK speaker-dependent system, Duchnowski has achieved a phone recognition accuracy of 79%. This system used right-context-dependent phone modeling, had a phone substitution rate of 11%, a phone deletion rate of 4% and a phone insertion rate of 6% [11].

1.2 Objective

The Sensory Communication Group of RLE is taking advantage of the recent advancements in the ASR field to design an automatic cued speech system to be used by the deaf in controlled settings such as classrooms and homes. The proposed system will use a computer to simultaneously display the speaker and cues. The objective of this study was to evaluate the effect ASR errors have on Cued Speech reception. Synthetic cues were superimposed on prerecorded sentences and simulated errors consisting of substitutions, deletions and insertions, as well as delayed cue presentation were introduced. The synthetic cues resembled those of Manual Cued Speech so that experienced, deaf Cued Speech receivers could be used as subjects with a minimal amount of training. Subjects were tested on the synthetic cued speech materials and the results were compared with their speechreading alone and Manual Cued Speech scores. If a significant improvement over speechreading alone is achieved with the synthetic cued speech that includes errors, then building a real-time automatic cued speech system would be beneficial. The results may also suggest areas for improving the ASR by providing insight into how much error is tolerable and what types of errors effect speechreading scores the most.
Chapter 2

Methods

2.1 Speech Materials

Three different types of sentences were used in the study: CID [12], Clarke [13] and Harvard [14]. The sentences were spoken by the same female teacher of the deaf (SR) with and without cueing. Three versions of the sentences were simultaneously recorded onto videotape, along with a running SMPTE time code. The first version was an audiovisual signal using a wide angle camera lens (the face filled up the top half of the vertical viewing area and roughly three-eighthes of the horizontal viewing area). The second version was an audiovisual signal using a narrow angle camera lens (the face filled up the top five-sixths of the vertical viewing area and nine-sixteenths of the horizontal viewing area). The last version was a PCM version, in which just the audio signal was recorded onto the videotrack of the tape.

With the aid of the SMPTE time code on window dub tapes, the beginning and end of the visual and audio portion of each sentence were noted. The audio tracks of the sentences from the PCM tapes were digitized and a phonetic transcript for each sentence, with corresponding time code markings, was produced using a Sensemetrics Speech Station. The sentences were transferred to laserdisc using the beginning and end times for the visual portion of the sentence.
2.1.1 CID Sentences

There are ten lists of CID sentences with each list containing ten sentences. Each sentence contains three to eight key words, with a total of 49-51 key words per list. The CID sentences are comparatively easy to speechread with subject matter consisting of everyday, conversational topics. Example sentences include ‘Walking’s my favorite exercise.’, ‘Do you want an egg for breakfast?’, and ‘Everything’s all right.’ All 100 sentences, in the wide angle view without cueing, were used in training.

2.1.2 Clarke Sentences

There are 600 Clarke sentences with five to ten words per sentence (no key words are indicated). The Clarke sentences are somewhat more difficult to speechread than the CID sentences with subject matter consisting of conversational topics arranged in a predictable manner. Example sentences include ‘Seven boys made a long table at school.’, ‘Miss Brown liked to read about animals.’, and ‘The house behind the hospital was old.’ The wide angle, noncueing versions of sentences 1–30, 41–50, 61–140, and 151–230 were used in training, along with the wide angle, cueing versions of sentences 401–420.

2.1.3 Harvard Sentences

There are 72 lists of Harvard sentences with each list containing ten sentences. Each sentence contains five key words for a total of 50 key words per list. The Harvard sentences are more difficult than the CID and Clarke sentences because they have fewer contextual cues. Example Harvard sentences include ‘It’s easy to tell the depth of a well.’, ‘Do that with a wooden stick.’, and ‘The sky that morning was clear and bright blue.’ Only the narrow angle versions of the sentences without cueing were transferred to videodisc and used in testing. Only 15 lists of the wide angle versions of Cued sentences were transferred and used.
2.2 Recording of Synthetic Cues

To investigate the effect imperfect cues have on the reception of cued speech, synthetic cues were superimposed on the speech materials and recorded onto laserdisc. The synthetic cues differed from manual cues in two ways. First, the synthetic cues were presented discretely rather than continuously as in Manual Cued Speech so transitions were absent. Second, the synthetic cues were presented at a faster rate than in Manual Cued Speech (150 WPM vs 100 WPM) [4].

2.2.1 The Cues

The same handshapes used in Manual Cued Speech were used for the synthetic cues. Truevision’s Nuvista software for the Macintosh computer was used to capture images of the eight handshapes from the cued speech sentence materials on laserdisc. The images were then edited and reduced in Adobe’s Photoshop. Each handshape was saved in 32 bit PICT color format and was approximately 48kb in size.

2.2.2 The Playlists

The program KeMIX [15] was used to superimpose the handshapes onto the video sentence material so that the presentation of the cues and the movement of the lips were properly synchronized. The program took as input a file in the following format: 1) A position list containing screen coordinates\(^1\) for the vowel positions (chin, throat, mouth, side-front and side-down). Each handshape had its own set of coordinates for each vowel position; 2) An image list specifying the names of the handshape files; 3) A sentence list containing a sentence identifier followed by the sentence’s start and stop frames on videodisc; 4) A playlist containing a sentence identifier followed by a start and stop frame for a particular cue followed by the position and handshape for that cue (see Appendix D).

\(^1\)There were four different coordinate lists: one for the CID sentences, one for the Clarke sentences, one for Harvard lists 1–58, and one for Harvard lists 59–72.
2.2.3 Converting Times to Frames

The phonetic transcriptions for the sentences specified a time in milliseconds at which each phone acoustically began as well as a symbol indicating the role each phone played in the sentence, i.e. if the phone was the beginning, middle, or end of a syllable. The program State was written in C to convert the phonetic transcriptions into the playlist file format described above. The program took as input a transcription file and a file containing the sentence name with its start and stop frames on videodisc. The output was the playlist. One important task that the State program accomplished was converting the phone start times in milliseconds to laserdisc frames. The speech was digitized at 10000 samples per second from the PCM tapes. However, the Dyaxis Digital Audio Processor used to digitize the sentences over-sampled the speech, so all times had to be multiplied by a factor of 0.964. In addition, the acoustic waveform on the PCM tapes lagged the acoustic signal on the other tapes by approximately 530 msec. This value was subtracted from all the times prior to multiplying by the Dyaxis correction factor. Finally, millisecond times were changed to laserdisc frames and rounded to the nearest frame by the conversion factor of 30 frames per second. Converting start times to video frame numbers reduced their time resolution from a tenth of a millisecond to 33msec. The start times listed in the transcription files were more accurate than times produced by ASRs which are accurate to 10msec. Thus, the resolution lost by converting to frame numbers was probably insignificant.

2.2.4 Converting Phones to Cues

Manual Cued Speech assumes that consonants are typically followed by vowels and encodes each CV combination as a handshape at a specified position. MCS includes special rules for other combinations such as VC, C, CC and CVC. Based on these rules, State converted the phones to cue positions and handshapes with the use of the special symbols associated with the phonetic labels in the transcription files. The five symbols were: (')-a consonant preceding a vowel at the beginning of a syllable or a vowel at the beginning of a syllable; (,)-a vowel in the middle of a syllable;
(·)—a consonant preceding another consonant at the beginning of a syllable; (*)—a consonant at the end of a syllable; (+)—the release of a stop consonant. The special symbols helped to determine to which consonant/vowel combination a phone belonged (i.e., CV, VC, C, etc.) so that the appropriate MCS rule could be applied. For example, when State encountered a consonant phone label marked by an apostrophe (‘) followed by a vowel phone marked by a comma (,) the corresponding consonant was assigned a handshape and the vowel was assigned a position via a lookup routine. As another example, the MCS rule for a consonant occurring by itself is that it should be cued at the side position, so when State encountered consonant phone marked by an asterisk (*), it automatically placed the assigned handshape at the side position.

Once phones were assigned handshapes and positions, State determined the start and stop frames for the cues. Appearance of the cue typically began at the consonant start frame and remained in a fixed position until the end of the consonant (for consonants occurring alone) or until the end of the proceeding vowel (for CV combinations). CV combinations that included a diphthong divided the cue duration evenly between the two vowels of the diphthong.

Different versions of State were written to produce the various error conditions discussed below. Also, a later version of State was written that did not require the syllable symbols but instead could process output directly from the ASR.

### 2.2.5 Recording

A block diagram of the system used to record the synthetic cues superimposed on the sentence materials is shown in Figure 2-1. Digital’s VAXstation 4000 (Hooper), running the KeMIX, controlled the Sony LVA-3500 Laser Videodisc player (in remote setting), the Sony LVR-5000 Laser Videodisc Processor/Recorder (in remote setting) and the Macintosh Quadra800. Truevision’s Nuvista videoboard and software, a peripheral to the Quadra800, coordinated the actual superposition of the handshapes onto the video.
2.3 Subjects

Four subjects, between the ages of 21–27, were employed in Phase I and five subjects between the ages of 19–27 in Phase II (see Table 2.1). All subjects were experienced cue receivers, having used Cued Speech for the past 12–23 years. All were native English speakers who were profoundly, prelingually deafened. No sound was presented during testing, so specific hearing losses are irrelevant. Their use of Cued Speech at the time of the study ranged from 1–10 hours per day, usually with a parent or transliterator.

Three of the six subjects (S1–S3) were tested in both phases of the study. S4
was tested only during Phase I, while S5 and S6 were tested only during Phase II. Subjects S1, S2, S5 and S6 had previous exposure to the speaker through school and prior testing. S3 and S4 had no previous exposure to the speaker.

2.4 Experimental Conditions

The experimental conditions in Phase I were chosen to simulate the type of errors an automatic speech recognizer might make. Substitution and deletion errors were based on the confusion matrix of a discrete HMM phonetic recognizer using context-independent modeling of 49 phones (including deletions, epinthetic and normal silences). Recognizer accuracy was 54% correct when tested on sentences from the TIMIT database [16]. The labeled phones in the sentence materials were slightly different from those of the HMM (phones /ix/ and /el/, as well as epinthetic and normal silences, were not labeled in the sentences). Inconsistencies between phone labeling conventions were rectified by editing the recognizer’s confusion matrix so that it only contained the phones that were labeled in the sentences. The confusion matrix CM1 was derived from the recognizer’s confusion matrix by converting percents to numbers of phones recognized and merging phone groups (/ix/ was merged with /ih/ and /el/ was merged with /l/). The silences were proportionally distributed to incorrectly recognized phones (e.g. if 3% of the errors were contained in the matrix entry '/ih/ input, /uh/ output', then 3% of the silences were added to that entry). The overall error rate of CM1 was the same as the recognizer’s matrix, 54%, but the distribution of the phones was changed (see Appendix E). The deletion rate for CM1 was 9% and the substitution rate was 37%.

Based on the results of Phase I, the conditions for Phase II were chosen to further investigate the effect of errors. In both phases, a short training session, using the CID and Clarke sentences, was conducted prior to each testing condition. Feedback was provided on the training sentences and all subjects received the same training sentences for each condition.

The Harvard sentences were used for testing and no feedback was provided. Be-
cause there is no measure of cueing difficulty for the Harvard sentences, it is impossible to predict if one list is equivalent to another. To alleviate this problem, the set of lists used under each condition was varied from subject to subject. Thus, no two subjects (except S2 and S6 in Phase II) saw the same sentence for a single condition.

Each Harvard sentence was strictly\textsuperscript{3} scored on five key words. Each key word is either correct or incorrect, and thus can be modeled as a Bernoulli random variable. Since the Harvard sentences provide few contextual cues, the key words can be considered to be statistically independent, which means the total number of correct key words can be modeled as a binomial random variable with expected value $Np$ and variance $Np(1 - p)$, where $N$ is the total number of key words and $p$ is the probability of correctly identifying a key word. Choosing a probability of 0.84 for the binomial random variable is consistent with results reported by Uchanski et al. [4] of 84% correct on the Harvard sentences in the Manual Cued Speech condition. Since the results are recorded as the fraction of correct key words out of the total number of key words, the expected value for the binomial random variable was divided by $N$ and the variance by $N^2$. Roughly 95% of the binomial distribution falls within two standard deviations of the expected value. To limit this two sigma spread to 5 percentage points the above formula for variance is solved for $N$. 215 key words are needed to achieve a standard deviation of 2.5%. Thus, each condition was tested using four lists (200 key words).

\subsection{Phase I}

There were eight experimental conditions for Phase I: speechreading alone (SA), Manual Cued Speech (MCS), perfect synthetic cued speech (PSC), synthetic cued speech with 20% errors (SD1), synthetic cued speech with 10% errors (SD2), synthetic cued speech with 20% errors and one frame of delay (D1), synthetic cued speech with 20% errors and three frames of delay (D3), and synthetic cued speech with 10% errors and

\textsuperscript{2}Some lists were seen by multiple subjects during the speechreading alone and Manual Cued Speech conditions.

\textsuperscript{3}Strict in the sense that plural nouns were not accepted for singular, past tense was not accepted for present, etc.
Table 2.2: Experimental Conditions for Phase I.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Error Rate(%)</th>
<th>Delay (frames)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MCS</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PSC</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SD1</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>SD2</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>D1</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>D3</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>D5</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>

five frames of delay (D5) (see Table 2.2). All conditions were presented on a video monitor without the audio signal.

SA and MCS

The speechreading alone and Manual Cued Speech conditions were used as baseline measures for comparison with the synthetic cue conditions. The sentence materials were recorded onto the laserdiscs without any processing.

PSC

The production of the perfect synthetic cued speech sentences consisted of converting the phonetic transcripts into playlist files using the State program without introducing errors or delays. The condition was perfect in that it was an exact conversion from the phonetic transcriptions. However, native English speakers tend to drop some phones and slur others together. Since the transcriptions came from acoustic waveforms, rather than the sentence orthography, some phonetic events were missing and some were confused.

SD1 and SD2

SD1 and SD2 incorporated 20% and 10% errors respectively without delay. The errors consisted of substitutions and deletions consistent with the errors of the CM1 confu-
sion matrix. Production of the playlists involved two steps. The phonetic transcriptions were processed by the program S2 [17] which substituted and deleted phonemes at a specified rate (i.e. 20% and 10%). The error rate was achieved by increasing the number of correct identifications for each input phone in CM1 while leaving the number of errors unchanged. For each phone in the transcription file, a non-zero entry from the corresponding row of the resulting matrix was randomly chosen without replacement. The selected phone replaced the correct phone in a new transcription file (*.lgo). A file (*.lgp) identifying the error phones along with the correct phones to the right was the second output of S2. The new phonetic transcriptions were processed by State which outputed the playlist files (see Appendix D).

D1, D3 and D5

D1, D3 and D5, which contained 20%, 20% and 10% errors respectively, also incurred one, three and five frame delays (33, 100 and 165 msec). New transcription files containing errors were produced using S2, the same as described for conditions SD1 and SD2. Playlist files were produced by processing the new phonetic transcriptions with delay versions of the State program. The delay versions added the appropriate frame delay to each start time immediately after they had been converted to frame numbers.

2.4.2 Phase II

There were nine experimental conditions (six new conditions) for Phase II: speechreading alone (SA), Manual Cued Speech (MCS), perfect synthetic cued speech (PSC), perfect synthetic cued speech with random frame delay (RD0), synthetic cued speech with 20% errors and random frame delay (RD1), synthetic cued speech with 10% errors and random frame delay (RD2), synthetic cued speech with 20% errors including insertions (INS), synthetic cued speech with 10% unmarked errors, 10% marked errors and 10% non-errors marked as errors (MARK), and synthetic cued speech with sentences processed by a phonetic automatic speech recognizer (REAL) (see Table 2.3).
Table 2.3: Experimental Conditions for Phase II.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Error Rate(%)</th>
<th>Delay (frames)</th>
<th>Special Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>MCS</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PSC</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>RD0</td>
<td>0 +/-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD1</td>
<td>20 +/-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD2</td>
<td>10 +/-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INS</td>
<td>20 0</td>
<td>includes insertions</td>
<td></td>
</tr>
<tr>
<td>MARK</td>
<td>20 0</td>
<td>some cues marked as errors</td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td>21 0</td>
<td>output of phonetic recognizer</td>
<td></td>
</tr>
</tbody>
</table>

All conditions with errors had substitution and deletions errors, except conditions INS and REAL which also had insertion errors.

**RD0, RD1 and RD2**

RD0, RD1 and RD2, which contained 0%, 20% and 10% errors respectively, also incurred random delay on a per phone basis. The random advance and delay was suppose to simulate the jitter that may occur in a real-time system. The phonetic transcriptions were processed by S2, as previously described. The new phonetic transcriptions were processed by the random delay version of State. After the start times were converted to frame numbers, a 1 frame advance, 1 frame delay or no delay was randomly chosen for each phone.

**INS**

Because the CM1 confusion matrix contained no information on insertion errors, the insertions were generated randomly. The sum of deletions and insertions often remains constant for a particular recognizer [18]. CM1 was adapted to contain insertions based on this principle. The number of deletions for each input phone was halved. The total number of removed deletions was evenly distributed to each input phone as insertions. The phonetic transcriptions were processed by S2 using the confusion matrix CM2.
edited to include insertions (see Appendix E). When S2 randomly chose an insertion to occur, the correct phone was replaced with the insertion symbol ‘ooo’.

The Stateins program was used to process the *.lgp files to output yet another new transcription file in which the ‘ooo’s were replaced with phones. The correct phone, listed to the right in the *.lgp file, was placed back into the transcription immediately before the insertion symbol. Next, a phone was randomly chosen to replace the insertion symbol. Finally, the length of the inserted phone was looked up in a table and room was made for it by shortening the length of the phones before and after it\(^4\). The ‘pre-insert’ phone was shortened by calculating the fraction:

\[
\text{‘pre-insert’ phone length} / (\text{‘pre-insert’ phone length} + \text{‘post-insert’ phone length}).
\]

The inserted phone length was multiplied by this fraction and the result was subtracted from the ‘pre-insert’ phone length. The ‘post-insert’ phone was shortened in a similar manner. For example, if the ‘pre-insert’ phone was 193msec, the ‘post-insert’ phone 82msec and the inserted phone 48msec, then 33.7msec \((48 \times 193/275)\) was subtracted from the end of the ‘pre-insert’ phone and 14.3msec \((48 \times 82/275)\) was subtracted from the beginning of the ‘post-insert’ phone. However, if shortening either the preceding or proceeding phone caused it to be shorter than 20msec, then only the longer phone was shortened. In the previous example, if the ‘post-insert’ phone was 25msec, shortening it would cause it to become 19.5msec. In this case all 48msec would be subtracted from the ‘pre-insert’ phone. If both pre and post phones were shortened to less than 20msec, the inserted phone was reduced by 20% until room could be made without any phone having a length less than 20msec.

Once all the insertion symbols in the transcription file were replaced with phones, a playlist was produced using the State.

\(^4\)The lengths of the inserted phones were 30% shorter than the average phones. The averages were calculated over 718 Harvard sentences for SR. If the average phone length was shorter than 20msec, the inserted phone was not reduced.
MARK

The phonetic transcripts of the MARK condition were processed by S2 to produce transcription files with 20% substitution and deletion errors. The *.lgp files were processed by a the mark version of the State program to mark half the errors (10% of the total phones). The same number (10% of the total) of correct phones were marked as well. The marked phones were chosen randomly and received red box outlines around their handshapes to indicate to the subject that they might be wrong.

REAL

The acoustic signal of the sentences in this condition were processed directly by Paul Duchnowski using a continuous Hidden Markov Model phonetic speech recognizer [19, 20] implemented using the Entropic HTK software [21, 9]. 25 features were extracted every 10msec from the acoustic signal (12 mel-frequency cepstral coefficients, 12 delta coefficients and one measure of normalized energy, all taken from a 20msec frame). The features were divided into a static vector and a dynamic vector (The two vectors were assumed to be statistically independent). The distribution of each feature vector was modeled with a mixture of six Gaussians. The recognizer was trained using 960 sentences (a mixture of CID, Clarke and Harvard) spoken by SR, from which 1151 right-context dependent phone models were derived. The models were three state, no skip HMMs, with the first state tied across contexts. A simple bigram language model and a Viterbi search were used to carry out the recognition. The recognition accuracy was 79.11%. 180 test sentences were processed. The output of the speech recognizer was converted by hand to phonetic transcription form and processed by the State to yield playlists.

2.5 Procedure

Four lists (40 sentences) of Harvard sentences were used in testing each condition. A combination of 20 CID and Clarke sentences were used in training each condition, except as noted for conditions SA, MCS and PSC. No sentence was presented more
than once to a single subject in either phase.

The subjects were seated inside a sound-treated booth, except for S4 who was tested at home. The subjects were tested in separate booths or at separate times (except S2 and S6 in Phase II, who were tested together). The subjects attended to a video monitor, which was roughly four feet away. The laserdisc player (VCR for S4) was paused after each sentence presentation to allow the subjects time to write their responses on a sheet of paper. The pause button was under the control of a tester(s) who waited until the subjects were ready before continuing. During training, the tester(s) repeated the correct response to the subjects and showed them the written answer if they did not understand. All sentences were scored at a later time.

2.5.1 Phase I

Testing was done over two days for each subject. Frequent breaks were taken, usually after the completion of 60 test sentences. Conditions SA, MCS and PSC were presented first to attain baseline scores. Training on these three conditions was consecutive (with ten training sentences for SA and MCS, and 30 for PSC), followed by 20 testing sentences of each in the same order.

Each condition involving errors and delays (SD1, SD2, D1, D3, and D5) was broken into two blocks consisting of ten training sentences and two testing lists (20 sentences). It was assumed that some learning would occur over the course of testing. To keep the affect of learning to a minimum, all conditions were presented once before repeating conditions. In addition, the order in which the blocks were presented was randomized to help eliminate learning affects. The presentation order remained the same for all subjects.

Finally, two lists of the SA condition followed by two lists of the MCS condition were presented at the end (without training) to obtain a post-testing baseline score.
2.5.2 Phase II

Phase II testing was conducted approximately two months after Phase I (see Tables C.3 and C.4). Testing again was done over two days and frequent breaks were provided. Each condition was broken into two blocks consisting of ten training sentences and two testing lists. One block of each condition SA, MCS and PSC was presented at the beginning of testing and the other at the end. Training was only provided at the beginning for conditions SA, MCS and PSC (10 sentences for SA, 10 for MCS and 20 for PSC). The order or presentation for those three conditions was randomized. The blocks of the remaining conditions (RD0, RD1, RD2, INS, MARK, and REAL) were presented randomly. As in Phase I, all conditions were presented once before any condition was repeated and the order was kept the same for all subjects.
Chapter 3

Results

3.1 Data Analysis

3.1.1 Scoring

The Harvard test sentences were scored for accuracy on five key words per sentence. For each subject, list scores (10 sentences, 50 key words) were compiled as well as an average across lists for each condition. The CID training sentences were also scored for accuracy on 2–10 key words per sentence, for a total of 50 key words per list (10 sentences). The Clarke training sentences were scored for accuracy on every word in the sentence since no scoring standard exists. Sentence lengths for the Clarke sentences ranged from 5–10 words.

3.1.2 Statistical Analysis

An analysis of variance (ANOVA) method [22, 23, 24] was applied to the test results (Harvard sentences only) to determine which factors played a statistically significant role in the observed differences in scores, as well as which factors, if any, were statistically related. The analysis was conducted on each phase separately. It was assumed that there were no differences among lists. The two factors involved in the ANOVA were conditions and subjects. The conditions factor was assumed to be a fixed effect because the set of conditions chosen were the only ones of concern. The subject factor
Table 3.1: Average test scores for subjects in Phase I.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Error Rate(%)</th>
<th>Delay (frames)</th>
<th>Subject</th>
<th>Subject</th>
<th>Subject</th>
<th>Subject</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>33</td>
<td>20</td>
<td>15</td>
<td>24.75</td>
</tr>
<tr>
<td>MCS</td>
<td>0</td>
<td>0</td>
<td>87</td>
<td>91</td>
<td>82</td>
<td>54</td>
<td>78.50</td>
</tr>
<tr>
<td>PSC</td>
<td>0</td>
<td>0</td>
<td>79</td>
<td>75</td>
<td>74</td>
<td>63</td>
<td>72.75</td>
</tr>
<tr>
<td>SD1</td>
<td>20</td>
<td>0</td>
<td>63</td>
<td>57</td>
<td>43</td>
<td>32</td>
<td>48.75</td>
</tr>
<tr>
<td>SD2</td>
<td>10</td>
<td>0</td>
<td>70</td>
<td>72</td>
<td>55</td>
<td>38</td>
<td>58.75</td>
</tr>
<tr>
<td>D1</td>
<td>20</td>
<td>1</td>
<td>54</td>
<td>55</td>
<td>47</td>
<td>40</td>
<td>49.00</td>
</tr>
<tr>
<td>D3</td>
<td>20</td>
<td>3</td>
<td>46</td>
<td>39</td>
<td>36</td>
<td>28</td>
<td>37.25</td>
</tr>
<tr>
<td>D5</td>
<td>10</td>
<td>5</td>
<td>40</td>
<td>44</td>
<td>31</td>
<td>24</td>
<td>34.75</td>
</tr>
</tbody>
</table>

Scores are percentage points based on four lists (200 key words) for each condition, except for SA and MCS which are based on six lists (300 key words).

was also assumed to be a fixed effect because the subjects were not randomly chosen. The analysis was also performed assuming that the subjects factor was a random effect and the results were essentially unchanged.

An F test at the 0.01 significance level under the null hypothesis of $\sigma^2 = 0$ for the term under consideration was performed on: conditions, subjects, and conditions $\times$ subjects (see Appendix B for a more detailed description of the ANOVA and F test). According to the F tests, there were significant differences in scores between conditions and significant differences between subjects, but there was no significant interaction between conditions and subjects. In other words, the observed differences between conditions were independent of subject. These results held true in both phases.

Based on the results of the F tests, paired t-tests at the two-tail 0.01 significance level were performed for all possible pairings of conditions in a single phase (see Appendix B for a more detailed description of these t-tests). The results are discussed below.

3.2 Phase I

Average scores for each condition across the four subjects in Phase I are shown in Table 3.1. The speechreading alone (SA) scores for all subjects fell below 40%. Base
on Figure 1 of Uchanski et al., phoneme reception that yields 40% correct on Harvard sentences translate to almost 90% correct on everyday sentences such as the CID or Clarke sentences and is often taken as the minimal level of speech reception adequate for everyday communication [4]. Subjects scored significantly better on all conditions over speechreading alone, and scored above 40% on all conditions except D3 and D5 (see Figures 3-1 and 3-2). Manual Cued Speech (MCS) scores were significantly higher than all other conditions except perfect synthetic cues (PSC). The difference between MCS and PSC was not significant according to the t-test, even when the results for S4 (the only subject to score higher on condition PSC than MCS), were not considered. The difference between perfect synthetic cues and synthetic cues with 10% errors (SD2) was also not statistically significant. This result is encouraging because it suggests that an automatic cueing system with 10% errors could provide benefits to cue receivers almost as well as a perfect system. However, increasing the errors by 10 percentage points (SD1, 20% errors) causes a significant decrease in key word reception compared with PSC and SD2 (Figure 3-1).
The difference in scores between D1 and SD1, both containing 20% errors, is small and not statistically significant, suggesting the delay of 33msec did not bother subjects. As delay increases, test scores decrease (see Figure 3-2). This trend holds true even when the error rate is decreased to compensate for the longer delays, as in condition D5. All subjects remarked that the delay was noticeable and bothersome on condition D5, and some remarked on the delay during condition D3 as well. The difference in scores between conditions D1 and D3 (a difference of two frames) was significant, but the difference in scores between D3 and D5 (also a two frame difference) was not statistically significant. It is likely that any noticeable delay has a detrimental affect on speech reception.

3.3 Phase II

Average scores for each condition across the five subjects in Phase II are shown in Table 3.2. Most of the subjects scored at or below 40% for speechreading alone. Scores
Table 3.2: Average test scores for subjects in Phase II.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Error Rate(%)</th>
<th>Delay (frames)</th>
<th>Subject</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>S1</td>
<td>S2</td>
</tr>
<tr>
<td>SA</td>
<td>0</td>
<td>0</td>
<td>43</td>
<td>40</td>
</tr>
<tr>
<td>MCS</td>
<td>0</td>
<td>0</td>
<td>92</td>
<td>93</td>
</tr>
<tr>
<td>PSC</td>
<td>0</td>
<td>0</td>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>RD0</td>
<td>0</td>
<td>+/-1</td>
<td>74</td>
<td>87</td>
</tr>
<tr>
<td>RD1</td>
<td>20</td>
<td>+/-1</td>
<td>65</td>
<td>71</td>
</tr>
<tr>
<td>RD2</td>
<td>10</td>
<td>+/-1</td>
<td>71</td>
<td>80</td>
</tr>
<tr>
<td>MARK</td>
<td>20</td>
<td>0</td>
<td>61</td>
<td>61</td>
</tr>
<tr>
<td>INS</td>
<td>20</td>
<td>0</td>
<td>60</td>
<td>61</td>
</tr>
<tr>
<td>REAL</td>
<td>21</td>
<td>0</td>
<td>79</td>
<td>73</td>
</tr>
</tbody>
</table>

Scores are percentage points based on four lists (200 key words) for each condition.

on the other conditions were above 40% and significantly better than speechreading alone (see Figures 3-3, 3-4 and 3-5). The two-tail probability for the paired t-test between MCS and PSC in Phase II was 0.0135. This number is very close to the 0.01 significance comparison level. A similar t-test performed in Phase I yielded no significant difference between MCS and PSC. Also, a t-test performed on the average across both phases for MCS and PSC yielded a two-tail probability of 0.1358 when all subjects were included and a two-tail probability of 0.0304 when only the repeat subjects were included. Either way, MCS and PSC do not appear to be very different. This suggests that the discrete nature of the synthetic cues and the faster presentation of the synthetic cues does not have a large impact on the cue receiver.

The differences between MCS and RD0 (random delay, no errors) was not significant, nor was the difference between PSC and RD0 (scores for all other conditions were significantly lower than MCS). This result suggests that a random one frame (33msec) delay and advance of the cues with no errors is not perceivable and/or not bothersome. Figure 3-3 plots the three random delay conditions in relation to SA and PSC. Some of the subjects actually did better on RD0 than on PSC. PSC was presented five conditions before RD0, so it is possible that learning occurred to help boost the RD0 scores. S6, the subject with the largest score differential between the two conditions, only participated in Phase II and thus probably experienced greater
learning during the first half of Phase II than subjects who had seen the synthetic cues before. The difference between PSC and the other random delay conditions (RD1 and RD2) was significant. The difference between RD0 and RD1 was statistically significant but the difference between RD0 and RD2 was not. For the most part, as the error rate increased speech reception decreased.

Three conditions contained 20% errors: RD1, INS and MARK. Figure 3-4 plots them together along with SA, MCS and PSC. There was no statistical difference between any of the three conditions, although subjects tended to do better on RD1. All conditions were significantly lower than RD0, but still remained above the 40%. The only condition subjects expressed a dislike for was the MARK condition in which red boxes appeared around the hand signaling to the subject that the cue may be in error. Most subjects found the box distracting and tended to ignore it rather than use it to their advantage. However, S3 tried to concentrate more on the lips and less on the cues when the red boxes appeared.

The REAL condition was one of the most important tests conducted in Phase II
because it showed how well subjects performed using output from the actual speech recognizer. Although differences were not statistically significant, most subjects performed slightly better on the REAL condition than on RD1 and RD2. Condition INS was most similar to REAL because it was the only other condition containing insertions. The deletion rate for the INS condition was 2.2%, and the random insertion rate was 8.6%. The REAL condition had a higher deletion rate of 4% but a lower insertion rate of 6%. Figure 3-5 plots conditions REAL and INS, along with SA and PSC. Subjects scored significantly higher on the real recognizer output than on the INS condition. The insertions produced by the ASR were not random like those in condition INS, which may account for the difference in scores.

### 3.4 Comparisons across Phases

Three subjects (S1, S2, and S3) participated in both phases of the study. There were also three conditions (SA, MCS and PSC) repeated in both phases. The three
subjects scored significantly higher on most of these conditions in Phase II than in Phase I suggesting that some amount of learning occurred over the course of the study (see Table 3.3). On average, SA scores went up by six percentage points and the MCS–PSC difference dropped from 11 percentage points to six percentage points. Condition RD1 of Phase II compared with SD1 of Phase I, both with 20% errors, provides additional evidence that learning occurred. Although RD1 contained random delay (the only difference between the two conditions), scores were higher than SD1 for all subjects (see Figure 3-6).

Table 3.3: Conditions SA, MCS and PSC across both Phases.

<table>
<thead>
<tr>
<th>Condition</th>
<th>S1 Phase I</th>
<th>Phase II</th>
<th>S2 Phase I</th>
<th>Phase II</th>
<th>S3 Phase I</th>
<th>Phase II</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>31</td>
<td>43</td>
<td>33</td>
<td>40</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>MCS</td>
<td>87</td>
<td>92</td>
<td>91</td>
<td>93</td>
<td>82</td>
<td>86</td>
</tr>
<tr>
<td>PSC</td>
<td>79</td>
<td>85</td>
<td>75</td>
<td>85</td>
<td>74</td>
<td>83</td>
</tr>
</tbody>
</table>

Scores are percentage points based on four lists, except for SA and MCS in Phase I which are based on six lists.
Figure 3-6: Comparison of error conditions across phases.

Figure 3-7 plots one delay condition, D1, from Phase I and two delay conditions, RD1 and RD2, from Phase II. D1 contained 20% errors with a one frame delay, while RD1 and RD2 contained a random one frame delay and advance with 20% and 10% errors, respectively. Scores for RD1 were higher than for D1, suggesting that learning occurred to boost RD1 scores. It is also possible that the random delay and advance was less noticeable than constant delay.

Condition SD1 of Phase I and the INS condition of Phase II were similar because both had error rates of 20% and no delays. However, INS contained a quarter of the deletions of SD1, with the difference made up by insertions. The tradeoff between deletions and insertions only seemed to affect S3 suggesting that insertions are neither better nor worse than deletions (see Figure 3-6).

### 3.5 Phone Error Rates vs Key Word Error Rates

Although the phone error rates were nominally set at 10% (in conditions SD2, D5 and RD2) and 20% (in conditions SD1, D1, D5, RD1, MARK, and INS), the corresponding
error rates for key words were much higher. Table 3.4 lists the actual phone error rate (Ph) and the corresponding key word error rate (KW) for each subject in both phases. The phone error rate is the percentage of phones that were incorrect out of the total number of phones for a given condition, not just those contained in key words. The phone error rates included deletions and insertions, where applicable, with deletions making up about half the errors (However, the INS condition error rate was an average of 30%, with a third of those errors being deletions and insertions.). The key word error rate is the percentage of key words that contained at least one phone error out of 200 key words for a given condition. A key word was counted as an error if one or more of its phones was an error. All key words contained a minimum of three phones, which resulted in key word error rates being three to four times greater than the phone error rates (except for the REAL conditions). In addition, 95% of the phone errors caused cueing errors (incorrect hand shape or vowel position).

Incorrect key words were not always missed by subjects even though they contained phone errors. Although results varied from subject to subject and condition

Subject
Circles = RD2; Filled Diamonds = RD1; Filled Inverted Triangles = D1.

Figure 3-7: Comparison of delay conditions across phases.
Table 3.4: Phone Error Rate and Key Word Error Rate.

<table>
<thead>
<tr>
<th>Condition</th>
<th>S1 Ph</th>
<th>S2 Ph</th>
<th>S3 Ph</th>
<th>S4 Ph</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD1</td>
<td>22</td>
<td>23</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td>SD2</td>
<td>11</td>
<td>11</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>D1</td>
<td>23</td>
<td>24</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>D3</td>
<td>21</td>
<td>24</td>
<td>25</td>
<td>24</td>
</tr>
<tr>
<td>D5</td>
<td>11</td>
<td>12</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition</th>
<th>S1 KW</th>
<th>S2 KW</th>
<th>S3 KW</th>
<th>S4 KW</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD1</td>
<td>63</td>
<td>61</td>
<td>65</td>
<td>66</td>
</tr>
<tr>
<td>SD2</td>
<td>33</td>
<td>40</td>
<td>39</td>
<td>41</td>
</tr>
<tr>
<td>D1</td>
<td>67</td>
<td>63</td>
<td>58</td>
<td>65</td>
</tr>
<tr>
<td>D3</td>
<td>55</td>
<td>61</td>
<td>66</td>
<td>67</td>
</tr>
<tr>
<td>D5</td>
<td>40</td>
<td>36</td>
<td>37</td>
<td>42</td>
</tr>
</tbody>
</table>

Phase II

<table>
<thead>
<tr>
<th>Condition</th>
<th>S1 Ph</th>
<th>S2 &amp; S6 Ph</th>
<th>S3 Ph</th>
<th>S5 Ph</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD1</td>
<td>23</td>
<td>23</td>
<td>24</td>
<td>23</td>
</tr>
<tr>
<td>RD2</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>MARK</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>INS</td>
<td>29</td>
<td>30</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>REAL</td>
<td>19</td>
<td>21</td>
<td>20</td>
<td>23</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition</th>
<th>S1 KW</th>
<th>S2 &amp; S6 KW</th>
<th>S3 KW</th>
<th>S5 KW</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD1</td>
<td>63</td>
<td>63</td>
<td>63</td>
<td>61</td>
</tr>
<tr>
<td>RD2</td>
<td>31</td>
<td>40</td>
<td>40</td>
<td>35</td>
</tr>
<tr>
<td>MARK</td>
<td>64</td>
<td>62</td>
<td>66</td>
<td>63</td>
</tr>
<tr>
<td>INS</td>
<td>68</td>
<td>76</td>
<td>75</td>
<td>77</td>
</tr>
<tr>
<td>REAL</td>
<td>34</td>
<td>38</td>
<td>39</td>
<td>36</td>
</tr>
</tbody>
</table>

Ph = Phone Error Rate; KW = Key Word Error Rate.

to condition, the percent of key words containing errors that subjects got correct was often above 50% for all conditions except D3 and D5 (see Table 3.5). The percent of key words containing errors that subjects got correct on those conditions was much lower, probably due to delay more than to errors. There are no clear trends in the results, but subjects tended to identify correctly a greater percentage of key words containing errors on conditions with lower overall error rates, such as SD2 and RD2 (Figures 3-8 and 3-9). In Phase II, the MARK condition had the lowest percentage of key words containing errors identified correctly which is consistent with Figure 3-4 and the distracting nature of the red boxes around the cues.

The existence of erroneous phones tended to influence the identification of words that did not contain phone errors (Table 3.6). For the most part, the percentage of key words containing no errors that subjects got correct increased as error rate decreased (Figures 3-8 and 3-9). Subjects' abilities to correctly identify error-free key words may have been affected by errors occurring in other key words as well as errors in surrounding function words (e.g. with, at, the etc.). The MARK condition did
Table 3.5: Percent of key words containing errors correctly identified.

### Phase I

<table>
<thead>
<tr>
<th>Subject</th>
<th>SD1</th>
<th>SD2</th>
<th>D1</th>
<th>D3</th>
<th>D5</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>60</td>
<td>66</td>
<td>52</td>
<td>34</td>
<td>43</td>
</tr>
<tr>
<td>S2</td>
<td>53</td>
<td>60</td>
<td>51</td>
<td>31</td>
<td>37</td>
</tr>
<tr>
<td>S3</td>
<td>38</td>
<td>47</td>
<td>44</td>
<td>34</td>
<td>24</td>
</tr>
<tr>
<td>S4</td>
<td>24</td>
<td>32</td>
<td>34</td>
<td>19</td>
<td>26</td>
</tr>
<tr>
<td>Average</td>
<td>43.75</td>
<td>51.25</td>
<td>45.25</td>
<td>29.50</td>
<td>32.50</td>
</tr>
</tbody>
</table>

### Phase II

<table>
<thead>
<tr>
<th>Subject</th>
<th>RD1</th>
<th>RD2</th>
<th>MARK</th>
<th>INS</th>
<th>REAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>61</td>
<td>60</td>
<td>58</td>
<td>44</td>
<td>68</td>
</tr>
<tr>
<td>S2</td>
<td>64</td>
<td>72</td>
<td>53</td>
<td>58</td>
<td>62</td>
</tr>
<tr>
<td>S3</td>
<td>39</td>
<td>59</td>
<td>43</td>
<td>61</td>
<td>61</td>
</tr>
<tr>
<td>S5</td>
<td>39</td>
<td>60</td>
<td>35</td>
<td>47</td>
<td>53</td>
</tr>
<tr>
<td>S6</td>
<td>66</td>
<td>63</td>
<td>52</td>
<td>56</td>
<td>47</td>
</tr>
<tr>
<td>Average</td>
<td>53.8</td>
<td>62.8</td>
<td>48.2</td>
<td>53.2</td>
<td>58.2</td>
</tr>
</tbody>
</table>

Percents calculated by dividing the number of key words with errors that were correct by the total number of key words with errors.

Table 3.6: Percent of key words containing no errors correctly identified.

### Phase I

<table>
<thead>
<tr>
<th>Subject</th>
<th>PSC</th>
<th>SD1</th>
<th>SD2</th>
<th>D1</th>
<th>D3</th>
<th>D5</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>79</td>
<td>68</td>
<td>73</td>
<td>57</td>
<td>61</td>
<td>38</td>
</tr>
<tr>
<td>S2</td>
<td>75</td>
<td>66</td>
<td>80</td>
<td>62</td>
<td>50</td>
<td>48</td>
</tr>
<tr>
<td>S3</td>
<td>74</td>
<td>53</td>
<td>61</td>
<td>52</td>
<td>41</td>
<td>35</td>
</tr>
<tr>
<td>S4</td>
<td>63</td>
<td>48</td>
<td>43</td>
<td>52</td>
<td>47</td>
<td>23</td>
</tr>
<tr>
<td>Average</td>
<td>73.25</td>
<td>58.75</td>
<td>64.25</td>
<td>55.75</td>
<td>49.75</td>
<td>36.00</td>
</tr>
</tbody>
</table>

### Phase II

<table>
<thead>
<tr>
<th>Subject</th>
<th>PSC</th>
<th>RD0</th>
<th>RD1</th>
<th>RD2</th>
<th>MARK</th>
<th>INS</th>
<th>REAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>85</td>
<td>74</td>
<td>73</td>
<td>76</td>
<td>68</td>
<td>82</td>
<td>79</td>
</tr>
<tr>
<td>S2</td>
<td>85</td>
<td>87</td>
<td>82</td>
<td>85</td>
<td>76</td>
<td>71</td>
<td>81</td>
</tr>
<tr>
<td>S3</td>
<td>83</td>
<td>65</td>
<td>77</td>
<td>72</td>
<td>59</td>
<td>56</td>
<td>78</td>
</tr>
<tr>
<td>S5</td>
<td>76</td>
<td>67</td>
<td>53</td>
<td>61</td>
<td>63</td>
<td>72</td>
<td>73</td>
</tr>
<tr>
<td>S6</td>
<td>78</td>
<td>88</td>
<td>78</td>
<td>73</td>
<td>76</td>
<td>67</td>
<td>75</td>
</tr>
<tr>
<td>Average</td>
<td>81.8</td>
<td>76.3</td>
<td>72.6</td>
<td>73.4</td>
<td>68.4</td>
<td>69.6</td>
<td>77.2</td>
</tr>
</tbody>
</table>

Percents calculated by dividing the number of key words with no errors that were correct by the total number of key words with no errors.
not have lower percentages than the other 20% error conditions, suggesting that the appearance of the red boxes did not affect the reception of error-free key words any more than the other conditions.

3.6 Comparison to Autocuer Study

It is difficult to compare the results presented here directly with those of the Autocuer study because the speech materials used for testing were vastly different (sentences verse isolated words). However, condition SD1 of Phase I is almost identical to conditions five and six of the Autocuer study because they both contain substitution and deletion type errors at a rate of 20%. Condition SD1 contained approximately 10% deletion and 10% substitution errors, while the Autocuer conditions contained 9% deletion and 11% substitution errors in condition five and 14% deletion and 6% substitution errors in condition six. Average scores on condition SD1 were 49% correct on the Harvard sentences, 80% on the Clarke sentences and 86% correct on the CID
Condition
Filled Circles: Average percentages for (correct key words/error-free key words);
Open Circles: Average percentages for (correct key words/erroneous key words).

Figure 3-9: Phase II.

sentences\(^1\). Average scores for the Autocuer study were 77% correct on condition five and 82% correct on condition six. Subjects in the Autocuer study did better on condition six with the higher deletion rate and lower insertion rate, suggesting that substitution errors have more effect on Cued Speech reception than deletion errors. The tradeoff between substitution and deletion errors was not explored in this study. On the everyday, conversational Clarke and CID sentences, subjects in Phase I scored the same or better than the Autocuer subjects on isolated words probably because of the highly contextual nature of the sentences. On the more difficult Harvard sentences, subjects scored far below the results seen in the Autocuer study. It is reasonable to assume that the isolated words in the Autocuer study were highly predictable since they came from a closed set of words and the average score for the words was close to scores seen on the highly predictable CID and Clarke sentences. Since 49% correct on the Harvard sentences translates to more than 90% correct on

\(^1\)The Clarke and CID average are only over subjects S1–S3 because S4 did not write down responses for the training sentences.
highly predictable words and sentences, subjects in this study performed better than subjects in the Autocuer study.
Chapter 4

Discussion

The results of Phase I and II, in conjunction with recent results from the HTK recognizer, confirm that building a real-time automatic cueing system would be beneficial to deaf cue receivers. Some suggestions, based on these results, for the design of a real-time system are given below. Discussion is restricted to an automatic cueing systems for use at home or in school that would display speakers and cues on a pc or laptop computer.

4.1 Effect of Synthetic Cues

The PSC condition, compared with Manual Cued Speech, provides a measure of how synthetic cues, with discrete handshapes and positions, affect cued speechreading. Subjects’ PSC scores were an average of 8 percentage points lower than MCS, a difference that was not statistically significant. Repeat subjects S1–S3 experienced more learning on the PSC condition than on MCS over the course of the two phases (12 hours total). Scores increased by an average of 8.3 percentage points on PSC verses 3.7 percentage points on MCS (see Table 3.3). It is possible that additional exposure to the synthetic cues might further increase scores.

The synthetic cues were different from the Manual Cues in four ways. First, presentation of the synthetic cues was 1.4 times faster on average than the Manual Cues. Most human cue producers reduce their speaking rates when cueing to roughly
100 WPM. The synthetic cues were superimposed onto uncued speech at rates similar to those a real-time system will encounter (140-180 WPM). Subjects commented on the increased speed when seeing the synthetic cues for the first time. It is most likely that cue producers reduce their speaking rates when cueing so that their hand can keep pace with their lips rather than for the benefit of the cue receiver. The effect of presentation rates on the reception of American Sign Language (ASL) key words was studied by Fischer, Delhorne and Reed using the CID sentences. The average score at the normal presentation rate (2.2 signs per second) was around 88% correct. Large reductions in scores were seen for presentation rates above two times normal. At an average rate 1.4 times faster than normal scores had only fallen to roughly 86% [25]. This suggest that a real-time cueing system operating at or slightly above normal speaking rates would not affect users' abilities to understand the cues greatly.

The second difference was that the synthetic cues were discrete in both shape and position and lacked transitions between cues. The synthetic cues could be made to look more continuous by interpolating handshapes at intermediate positions, but this would require an increase in storage. Furthermore, presenting synthetic cues continuously at normal speaking rates might blur the handshapes and make them harder to see. With discrete cues, the handshapes remain at the vowel positions longer, allowing the cue receiver to receive clear views of the handshape and vowel position. Only one subject (S6) complained about the lack of transitions, but this subject's PSC scores were close to MCS and may not have increased if continuous cues were available. Any increase in scores that continuous cues could elicit might be small and not worth the increase in memory.

Thirdly, the synthetic handshapes were smaller compared with the natural handshapes. This bothered many of the subjects, especially on the narrow-angle views. A smaller hand was used so that handshapes placed at the throat position on the narrow-angle view of the Harvard sentences (the only view available for those sentences on laserdisc) could be seen without being cut off at the bottom of the screen. Despite the smaller hands, subjects still had some difficulty distinguishing between cues at the throat position, as well as distinguishing between the throat and chin
positions. Subjects’ inabilities to distinguish cues at the throat may account for some of the difference in scores between PSC and MCS. The smaller handshape, displayed clearly at the other vowel positions, probably did not affect scores. A real-time system should leave the bottom quarter of the vertical display free for cue presentation as well as a third of the horizontal display. Using a video camera with an adjustable lens would allow speakers’ faces to be placed at the same coordinates on the display making it easier to properly align the handshapes at the vowel positions without obscuring the lips. Maintaining a constant face size, through the use of the adjustable lens, would make it possible to store only a single hand size for each shape.

The final difference, discussed in Section 2.4, was that the phonetic transcriptions of the PSC condition contained a small number of missing and incorrect phonetic events, while the MCS condition was perfect. The incorrect cues that did appear in the PSC condition contributed slightly to the difference between MCS and PSC scores.

The synthetic cues in this study were chosen to resemble the cues of Manual Cued Speech so that deaf subjects experienced in MCS could be used without an extensive amount of training. Despite the differences discussed above, subjects had little difficulty understanding the synthetic cues. The 1981 Autocuer study used a matrix of light-emitting diodes superimposed to the side of speech materials. The four quadrants of the matrix represented the vowel positions and the diode pattern the consonant handshapes [5, 6]. Alternative cue groups were also used. However, the technology available today makes it much easier to use the same handshapes and vowel positions of MCS in an automatic cueing system. As a supplement and teaching aid, a real-time system would be most useful if it used the same cues as MCS.

### 4.2 Effect of Error Rate

Ten conditions were used to explore the effect errors have on the reception of Cued Speech. Three conditions were assigned error rates of 10%, six conditions were assigned error rates of 20%, and the ASR error rate on the REAL condition was 21%.
Results from the REAL condition are most important because they provide a baseline measurement of how well a real-time cueing system might perform. The average score across all subjects of Phase II on the REAL condition was 70.2% correct (79% high, 64% low), which is only 10.2 percentage points below the average score for the PSC condition (in Phase II).

Most error rates were not exactly equal to the nominal ten or twenty percent figures because the method used to insert errors was random. S2, the program used to produce errors, adjusted the probability distribution of each phone in the CM1 confusion matrix so that the error rate was nominally 10% or 20% (see Section 2.4). Because each phone in a sentence was drawn randomly without replacement from the adjusted confusion matrix, it was impossible to guarantee the exact phone error rate on individual sentences. Post-testing calculations confirmed that most conditions had error rates close to the specified values (see Table 3.4).

Conditions SD1 and RD1 had average phone error rates of 23%, which is slightly higher than the error rate for the REAL condition. The average score for SD1 was 48.7% correct (63% high, 32% low) and for RD1 was 60.5% correct (71% high, 44% low). Although the three conditions had similar overall phone error rates, subjects performed much better on the REAL condition. The main differences between the simulated error conditions and REAL were the distribution of errors and key word error rates. SD1 and RD1 contained approximately 11% deletions and 11% substitutions, while REAL contained 4% deletions, 11% substitutions and 6% insertions. The high deletion rate of simulated conditions SD1 and RD1 may be the cause of the lower test scores. This result suggests that deletions have a greater effect on Cued Speech reception than insertions. Also, the key word error rate of REAL was half that of SD1 and RD1. With fewer key words in error, subjects were able to score higher on the REAL condition than on SD1 or RD1.

SD2 and RD2 contained approximately 6% deletions and 6% substitutions for overall error rates of 12%. Average correct scores for the two conditions were 58.7% (72% high, 38% low) and 69.4% (80% high, 60% low), respectively. The REAL condition outperformed SD2 and RD2 even though it contained twice as many phone
errors. The key word error rate of the REAL condition was similar to the key word error rates of SD2 and RD2, which helps to explain why the average scores were similar. However, it more difficult to explain why the key word error rates were similar. The errors of conditions SD2 and RD2 were derived from the CM1 matrix which was generated by a discrete HMM phonetic speech recognizer using context-independent phone models. Since each phone had an equal probability of being in error, the distribution of errors across a sentence is expected to be uniform. In contrast, the HTK phonetic recognizer used for the REAL condition used right-context dependent phone modeling. A cursory examination of error occurrence in the REAL sentences suggests that phone errors occurred in clusters rather than uniformly. Also, many errors occurred at the beginning and end of sentences. The error rate for phones in key words averaged 19.7% while the error rate for phones in non-key words (function words) averaged 23.7%\(^1\). The phone error rate for function words was only slightly higher than for the key words, so no definite conclusions can be drawn about the key word error rate based on these phone rates. Further examination of phone error clustering in key words to explain the low key word error rate of the REAL condition is suggested.

Two conditions had phone error rates higher than the nominal 20%. The MARK condition had an average phone error rate of 27%. Although this rate was slightly higher than the rates of SD1, RD1, D1 and D3, the 64% key word error rate for MARK was consistent with key word error rates on the other ‘20%’ conditions. Since the increase in phone error rate did not change the key word error rate, subjects’ scores were probably not affected either. There is no clear explanation for why the phone error rate increased without an increase in key word error rate.

The average key word score on the MARK condition was 55.4% (61% high, 46% low), which was 15 percentage points below the REAL condition. The difference in scores between MARK and REAL may lie in MARK’s higher deletion rate (around

\(^1\)The error rate for phones in key words was calculated by dividing the number of phone errors in key words by the total number of phones in key words. A similar calculation was made for the function words.
13%) and key word error rate, but it may also be due to the distracting nature of the red boxes used to signal possible erroneous cues. If subjects focus more on the lips and less on the cues as S3 described, then the cues outlined with the red boxes become like deleted cues because subjects are ignoring them. Providing feedback to users of an automatic cueing system could still prove to be useful, but it must be done in a such a way that it is not distracting. Also, it must not reduce the receiver's ability to keep pace with cue presentation. However, if errors occur mainly in non-essential words, as the results from the HTK recognizer suggest, then it may not be necessary to provide feedback to users about errors.

The other condition with an unexpectedly high phone error rate was the INS condition, with an average error rate of 30%. The key word error rate for INS also increased to around 75%, up from the 63% average key word error rate seen on the other '20%' conditions. The insertion column of confusion matrix CM2 contained non-zero probabilities in all cells, including the cells for closures (inspection of CM1 and CM2 in Appendix E shows that closures were never confused with other phones or deletions). Thus, when S2 encountered a closure in a transcription file, it was replaced with either itself, but more often an insertion. Because of the unusual distribution of errors in the insertion column of CM2, the insertion rate was four times greater than the deletion rate although the overall number of errors in the two columns were the same. The increased insertion rate may have been responsible for the increase in the phone error rate. In addition, inserted phones tended to be placed in the middle of key words causing a higher percent of key word errors. It is difficult to determine the effect of insertions since the phone error rate was higher for the INS condition. However, Figures 3-4 and 3-6 suggest that subjects did just as well on the INS condition as on conditions MARK, SD1 and RD1. The REAL and INS conditions have previously been compared in Section 3.3 and Figure 3-5. The difference in scores between REAL and INS can be easily explained by the difference in phone and key word error rates and the random nature of the errors in the INS condition.
4.3 Effect of Delay

In six conditions the visual cues were delayed relative to the start of the acoustic phone. In Phase I a constant delay of one, three and five frames (33, 100 and 165 msec) was added to conditions D1, D3 and D5, respectively. Figure 3-2 clearly shows that scores decreased as delay increased. A one frame delay (33msec) seemed tolerable since D1 scores were not significantly lower than SD1 scores (both with 20% errors). However, a dramatic drop in scores occurred with three frames of delay. Cue delay due to ASR processing time can be avoided in a real-time cueing system by delaying the presentation of the speaker’s face by the average processing time. Using the average processing time would cause some cues to be presented in advance of phoneme articulation and some cues to be presented after. If the variability of the processing time is less than 33msec (one frame, as investigated in Phase II with conditions RD0, RD1 and RD2), subjects will not be affected by the delay. Furthermore, scores in these three conditions are roughly comparable to PSC, SD1 and SD2, respectively. The average score for condition RD0, the only random delay condition without errors, was 76.2% correct (88% high, 65% low), only five percentage points below condition PSC. Some subjects even scored higher on condition RD0, probably due to practice effects (see Section 3.3). Average scores for RD1 and RD2 were higher than SD1 and SD2 because of learning across the two phases. Since the random delay was not noticeable, an average processing time delay could be incorporated into a real-time cueing system without affecting cue reception performance.

4.4 Choice of Cue Groups

Because each Manual Cued Speech group contains several phonemes, an ASR system need not distinguish among phonemes within the Cued Speech groups. Cornett chose the cue groups for Manual Cued Speech so that viseme confusions would be eliminated. However, such groups are not unique. Other cue groups may be better suited to ASR discrimination while still eliminating viseme confusions. In fact, Uchanski
et al. conducted a detailed, analytical study of the potential benefit that various cue groupings might have on speechreading scores. They found that 3–4 vowel cue groups and 3–4 consonant cues groups were needed to attain ‘near-maximum’ scores on cued speech [4]. These results were recognizer-dependent.

The phone errors in all conditions except REAL caused incorrect cue production 95% of the time. Similar results were seen in the key words of the REAL condition. The only phone substitutions that did not cause cue changes in the 160 test sentences of the REAL condition were the confusion pairs /l/-/w/, both in the l-handshape group, and /ax/-/ah/, both in the side-front vowel position group. Based on these results, it would seem that different cue groups could be formed to better suit the confusions made by the HTK recognizer (i.e. so that frequently confused phones fell into the same cue group). However, many of the HTK confusions fell within the same viseme groups (e.g. /p/ is confused with /b/). Placing them into the same cue group would defeat the purpose of Cued Speech which is to alleviate lipreading ambiguities of visemes.

Since many of the errors made by the HTK recognizer occur in words that are not crucial for understanding the meaning of the sentence, (i.e. function words) it may not be important to develop new cue groups. Cue receivers would require extensive training to learn the new groups and the groups would be inconsistent with Manual Cued Speech used at home and school.

4.5 Summary

Twelve synthetic cueing conditions were compared against speechreading alone and Manual Cued Speech. Subjects scored higher than speechreading alone on all conditions but were unable to achieve the high scores seen with Manual Cued Speech. The HTK recognizer outperformed all simulated conditions. Average scores on the REAL condition, which had a 21% phone error rate, were higher than the other error conditions because of the condition’s low key word error rate. One possible expla-

---

2This percent is based on all phones, not just ones in key words.
nation for the low key word error rate on the REAL condition is that phone errors were distributed nonuniformly. If this explanation is correct, then although the other error conditions were successful at simulating the error rate of ASRs, they failed to simulate the error distribution. In other words, where errors occurred was probably just as, if not more, important than how many occurred.

Deletions seemed to hurt speech reception more than insertions, thus greater tolerance should be given to insertions when developing ASRs. The feedback presented in this study to inform subjects about errors was not useful, but alternative forms of feedback should be explored. Cue delays of 33msec or less had little effect on speech reception.

At the end of the study the subjects were asked if they would use an imperfect cueing system, similar to what they'd seen during testing. All subjects responded that they would prefer an imperfect system to nothing at all.
Bibliography


Appendix A

Auxiliary Experiment: The Effect of Limited Hand Movements on the Reception of Cued Speech

A.1 Purpose

Many experienced cue producers, who cue at conversational speeds, tend not to move their hands to the exact vowel positions (mouth, chin, throat, side). Instead, they keep their hand at the side position and make slight movements towards the other vowel positions. This, in part, is what allows them to cue quickly.

An Auxiliary experiment was conducted to investigate the effect such limited hand movements have on the reception of Cued Speech. Because the vowels are easier to lip read than the consonants, vowel positions may not be as necessary as consonant shapes for speech reception.

A.2 Methods

The same recording process previously described in Section 2.2 was used to superimpose the synthetic handshapes onto Harvard sentences. There were four experimental conditions: no hand movements (NM), limited hand movements (LM), normal hand
movements (PSC) and speechreading alone (SA). None of the conditions included errors or delays. The side hand position coordinates were used to cue all phonemes in the NM condition. The LM hand position coordinates (Table A.2) were decided by Gabrielle Jones, an experienced Cue receiver with exposure to cue producers practicing limited hand movements. Miss Jones also recorded all the testing materials and was the only subject in the experiment.

The 21 year old subject was deafened at age 2 (unknown etiology), had been using Cued Speech since age 2 1/2, and at the time of the study was using Cued Speech 5–10 hours per day.

Four lists of Harvard sentences were used in each condition for testing. Each condition was broken into two sets of two lists. The sets were presented randomly (except for condition SA which was presented last). The first time a condition was presented it was preceded by one list of Harvard training sentences in which feedback was provided (no training was provided for condition SA).

A.3 Results

The average results for the four conditions were: 73% correct (NM), 86% correct (LM), 88.5% correct (PSC) and 57.5% correct (SA) (raw results are shown in Table A.1). All three Cued Speech conditions provided a significant boost to speechreading alone. The subject also scored much better on the conditions involving hand movements (LM and PSC) over the no hand movement condition. Condition LM was 13 percentage points higher than condition NM, and condition PSC was 15.5 percentage points higher. The difference between the limited hand movements and the normal hand movements was very small, only 2.5 percentage points.

A.4 Discussion

The results suggest that indicating the vowel positions is important in the reception of Cued Speech. Even limited hand movement allows the cue receiver to discriminate
Table A.1: Scores for Limited Hand Movement Study

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>NM</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>56</td>
<td>94</td>
</tr>
<tr>
<td>LM</td>
<td>74</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>84</td>
<td>74</td>
</tr>
<tr>
<td></td>
<td>96</td>
<td>98</td>
</tr>
<tr>
<td>PSC</td>
<td>76</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>74</td>
</tr>
<tr>
<td></td>
<td>98</td>
<td>68</td>
</tr>
<tr>
<td>SA</td>
<td>68</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>64</td>
</tr>
</tbody>
</table>

Scores are percentage points based on four lists (200 key words) for each condition.

between vowels and to achieve scores close to those of condition PSC. It should also be noted, however, that the subject was able to derive some benefit to speechreading when only the consonant handshapes were presented.

More data needs to be collected from other subjects before a solid conclusion can be drawn from the hand movement study. Also, it would be useful to look at other hand position coordinates which are more limited than those in this study.

Table A.2: Harvard sentences (59–72) position coordinates for LM.

<table>
<thead>
<tr>
<th>Cue Group</th>
<th>side</th>
<th>mouth</th>
<th>throat</th>
<th>chin</th>
<th>side-down</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>30</td>
<td>105</td>
<td>120</td>
<td>125</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>290</td>
<td>290</td>
<td>395</td>
<td>340</td>
<td>340</td>
</tr>
<tr>
<td>g</td>
<td>40</td>
<td>115</td>
<td>130</td>
<td>130</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>290</td>
<td>290</td>
<td>395</td>
<td>340</td>
<td>340</td>
</tr>
<tr>
<td>h</td>
<td>45</td>
<td>130</td>
<td>140</td>
<td>135</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>290</td>
<td>290</td>
<td>395</td>
<td>340</td>
<td>340</td>
</tr>
<tr>
<td>k</td>
<td>50</td>
<td>125</td>
<td>130</td>
<td>135</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>290</td>
<td>290</td>
<td>395</td>
<td>340</td>
<td>340</td>
</tr>
<tr>
<td>l</td>
<td>45</td>
<td>120</td>
<td>120</td>
<td>130</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>290</td>
<td>290</td>
<td>395</td>
<td>340</td>
<td>340</td>
</tr>
<tr>
<td>n</td>
<td>40</td>
<td>115</td>
<td>120</td>
<td>130</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>280</td>
<td>290</td>
<td>395</td>
<td>330</td>
<td>330</td>
</tr>
<tr>
<td>ng</td>
<td>45</td>
<td>115</td>
<td>130</td>
<td>125</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>260</td>
<td>260</td>
<td>395</td>
<td>310</td>
<td>310</td>
</tr>
<tr>
<td>t</td>
<td>30</td>
<td>105</td>
<td>120</td>
<td>115</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>280</td>
<td>280</td>
<td>395</td>
<td>320</td>
<td>320</td>
</tr>
</tbody>
</table>
Appendix B

Statistical Analysis

A two-way analysis of variance, F test and t-test are described below as they were applied to the results of Phase I and II. For further explanation the reader should refer to a standard text, such as Choi [22] or Dunn and Clark [23].

B.1 ANOVA

The goal of an analysis of variance (ANOVA) is to determine the factors and cross-factors that vary significantly from their sample means in an experiment. There were two factors, conditions and subjects, and one cross factor, conditions X subjects, in this study. The ANOVA also contains a ‘residual’ factor representing the variation of single observations from their sample means. A single observation in this study was a subject’s score on one list. The sum of squares (SS) of the differences between factor means and the overall mean represents the variance for a single factor. For example, the sum of squares for conditions is

$$sn \sum_{i=1}^{c}(\bar{Y}_{i.} - \bar{Y}_{..})^2$$

where the sum is taken over the number of conditions, c. For Phase I c = 8 and for Phase II c = 9. The left term in the sum represents the mean of one condition (e.g. the average score for condition SA across all subjects). The right term in the sum
Table B.1: Analysis of Variance for Condition-Subject Interaction

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>SS</th>
<th>DF</th>
<th>MS</th>
<th>EMS</th>
<th>Case 1</th>
<th>Case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Case 1</td>
<td>Case 2</td>
</tr>
<tr>
<td>conditions</td>
<td>17.71</td>
<td>7</td>
<td>2.53</td>
<td>$\sigma_e^2 + 16\sigma_c^2$</td>
<td>$\sigma_e^2 + 4\sigma_{cs}^2 + 16\sigma_c^2$</td>
<td></td>
</tr>
<tr>
<td>subjects</td>
<td>5.04</td>
<td>3</td>
<td>1.68</td>
<td>$\sigma_e^2 + 32\sigma_s^2$</td>
<td>$\sigma_e^2 + 32\sigma_s^2$</td>
<td></td>
</tr>
<tr>
<td>conditionsXsubjects</td>
<td>0.99</td>
<td>21</td>
<td>0.047</td>
<td>$\sigma_e^2 + 4\sigma_{cs}^2$</td>
<td>$\sigma_e^2 + 4\sigma_{cs}^2$</td>
<td></td>
</tr>
<tr>
<td>residual</td>
<td>4.46</td>
<td>96</td>
<td>0.046</td>
<td>$\sigma_e^2$</td>
<td>$\sigma_e^2$</td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>28.20</td>
<td>127</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>SS</th>
<th>DF</th>
<th>MS</th>
<th>EMS</th>
<th>fixed</th>
<th>variable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>fixed variable</td>
<td>variable</td>
</tr>
<tr>
<td>conditions</td>
<td>20.74</td>
<td>8</td>
<td>2.59</td>
<td>$\sigma_e^2 + 20\sigma_c^2$</td>
<td>$\sigma_e^2 + 4\sigma_{cs}^2 + 20\sigma_c^2$</td>
<td></td>
</tr>
<tr>
<td>subjects</td>
<td>2.67</td>
<td>4</td>
<td>0.667</td>
<td>$\sigma_e^2 + 36\sigma_s^2$</td>
<td>$\sigma_e^2 + 36\sigma_s^2$</td>
<td></td>
</tr>
<tr>
<td>conditionsXsubjects</td>
<td>1.85</td>
<td>32</td>
<td>0.058</td>
<td>$\sigma_e^2 + 4\sigma_{cs}^2$</td>
<td>$\sigma_e^2 + 4\sigma_{cs}^2$</td>
<td></td>
</tr>
<tr>
<td>residual</td>
<td>6.50</td>
<td>135</td>
<td>0.048</td>
<td>$\sigma_e^2$</td>
<td>$\sigma_e^2$</td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>31.76</td>
<td>179</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SS = sum of squares; DF = degrees of freedom; MS = mean square; EMS = expected mean square. Case 1 is subjects assumed fixed and case 2 is subjects assumed variable.

represents the overall mean of observations taken across all conditions and subjects.

The term outside the sum is the product of the number of subjects ($s = 4$ or 5) and the number of observations for a single condition ($n = 4$) [23]. Scores were arcsine-transformed to equalize the variance prior to taking the sum of squares.

The calculated mean square (MS) for each factor and cross-factor is obtained by dividing the sum of squares by the factor's degrees of freedom (DF). The sum of squares, degrees of freedoms and mean squares calculated from the results of Phase I and II are listed in Table B.1, along with the expected mean squares (EMS) derived in Dunn and Clark [23]. The expected mean squares for two cases are shown: 1.) subjects assumed to be a fixed effect and 2.) subjects assumed to be a variable effect.
Table B.2: F Tests.

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>F statistic Case 1</th>
<th>F statistic Case 2</th>
<th>reject/accept hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>conditions</td>
<td>$MS_c/\sigma_e^2 = 54.51$</td>
<td>$MS_c/MS_{cs} = 53.67$</td>
<td>reject</td>
</tr>
<tr>
<td>subjects</td>
<td>$MS_s/\sigma_e^2 = 36.23$</td>
<td>$MS_s/\sigma_e^2 = 36.23$</td>
<td>reject</td>
</tr>
<tr>
<td>conditionsXsubjects</td>
<td>$MS_{cs}/\sigma_e^2 = 1.01$</td>
<td>$MS_{cs}/\sigma_e^2 = 1.01$</td>
<td>accept</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>F statistic Case 1</th>
<th>F statistic Case 2</th>
<th>reject/accept hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>conditions</td>
<td>$MS_c/\sigma_e^2 = 53.85$</td>
<td>$MS_c/MS_{cs} = 44.76$</td>
<td>reject</td>
</tr>
<tr>
<td>subjects</td>
<td>$MS_s/\sigma_e^2 = 13.85$</td>
<td>$MS_s/\sigma_e^2 = 13.85$</td>
<td>reject</td>
</tr>
<tr>
<td>conditionsXsubjects</td>
<td>$MS_{cs}/\sigma_e^2 = 1.20$</td>
<td>$MS_{cs}/\sigma_e^2 = 1.20$</td>
<td>accept</td>
</tr>
</tbody>
</table>

**B.2 F Test**

The F test is used to decide whether to accept or reject the null hypothesis of $\sigma^2 = 0$. Acceptance means that a factor’s variance is not significantly greater than zero, while rejection means that a factor’s variance is significant. The F statistic is a ratio of two expected mean squares, chosen so that the statistic equals one when $\sigma^2 = 0$. The numerator of the F statistic is always the EMS for the factor being tested. The denominator is any other EMS that forces the F statistic to one. For example, the numerator for the conditions factor in Phase I (fixed case) is $\sigma_c^2 + 16\sigma_c^2$ and the denominator is $\sigma_e^2$. Under the null hypothesis, $\sigma_c^2 = 0$, so the F statistic is one. In numerically calculating the F statistic, mean squares are used in place of their corresponding EMSs. The F statistic of the previous example becomes $2.53/0.046 = 54.51$. The F test consists of comparing the calculated F statistic with a tabulated F statistic at a desired significance level [23]. A significance level of 0.01 was used in this study and tabulated F statistics from Ref [24]. Table B.2 lists the F statistics and the outcomes of the F tests. The factors (conditions and subjects) varied significantly from their means, but the cross-factor did not.
B.3 t-tests

If a factor's variance deviates from its mean significantly (based on the F test) then paired t-tests must be performed to identify the components contributing to the variation. In this study, paired t-tests were performed on the conditions factor to determine the pairs of conditions that varied significantly. T-tests were not performed on the subjects factor because differences between individual subjects were not of interest. An advantage of using the paired t-test is that the means being compared do not need to have equal variances.

To begin the t-test, the difference between a pair of condition means are taken for each subject. For example, the mean scores for condition SA are subtracted from the mean scores for condition MCS. Next, the overall mean, \( \bar{d} \), of the differences is calculated across all subjects, along with the standard deviation, \( s \). Finally the t-test statistic

\[
\frac{\bar{d}}{s/\sqrt{n}}
\]

(where \( n \) is the number of subjects) is compared with a tabulated value at the 0.01 significance level [22]. For this study a commercial statistical package [26] was used, the output of which was a two-tail probability. The probability represented the likelihood that the difference between two conditions was due to chance. Thus, high probabilities (close to one) indicated that it was very likely and that the difference between conditions was not significant, where as low probabilities indicated that it was not likely and that the difference was significant. Table B.3 lists the condition pairs and corresponding probabilities for both phases.
Table B.3: t-tests.

### Phase I

<table>
<thead>
<tr>
<th>Condition Pair</th>
<th>Probability</th>
<th>Significant?</th>
<th>Condition Pair</th>
<th>Probability</th>
<th>Significant?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA-MCS</td>
<td>.0019</td>
<td>yes</td>
<td>PSC-SD2</td>
<td>.0591</td>
<td>no</td>
</tr>
<tr>
<td>SA-PSC</td>
<td>.0003</td>
<td>yes</td>
<td>PSC-D1</td>
<td>.0003</td>
<td>yes</td>
</tr>
<tr>
<td>SA-SD1</td>
<td>.0043</td>
<td>yes</td>
<td>PSC-D3</td>
<td>.0001</td>
<td>yes</td>
</tr>
<tr>
<td>SA-SD2</td>
<td>.0027</td>
<td>yes</td>
<td>PSC-D5</td>
<td>.0006</td>
<td>yes</td>
</tr>
<tr>
<td>SA-D1</td>
<td>.0005</td>
<td>yes</td>
<td>SD1-SD2</td>
<td>.0143</td>
<td>yes</td>
</tr>
<tr>
<td>SA-D3</td>
<td>.0116</td>
<td>yes</td>
<td>SD1-D1</td>
<td>.5000</td>
<td>no</td>
</tr>
<tr>
<td>SA-D5</td>
<td>.0044</td>
<td>yes</td>
<td>SD1-D3</td>
<td>.0470</td>
<td>no</td>
</tr>
<tr>
<td>MCS-PSC</td>
<td>.3845</td>
<td>no</td>
<td>SD1-D5</td>
<td>.0187</td>
<td>no</td>
</tr>
<tr>
<td>MCS-SD1</td>
<td>.0051</td>
<td>yes</td>
<td>SD2-D1</td>
<td>.1152</td>
<td>no</td>
</tr>
<tr>
<td>MCS-SD2</td>
<td>.0044</td>
<td>yes</td>
<td>SD2-D3</td>
<td>.0186</td>
<td>no</td>
</tr>
<tr>
<td>MCS-D1</td>
<td>.0122</td>
<td>yes</td>
<td>SD2-D5</td>
<td>.0056</td>
<td>yes</td>
</tr>
<tr>
<td>MCS-D3</td>
<td>.0050</td>
<td>yes</td>
<td>D1-D3</td>
<td>.0035</td>
<td>yes</td>
</tr>
<tr>
<td>MCS-D5</td>
<td>.0026</td>
<td>yes</td>
<td>D1-D5</td>
<td>.0028</td>
<td>yes</td>
</tr>
<tr>
<td>PSC-SD1</td>
<td>.0094</td>
<td>yes</td>
<td>D3-D5</td>
<td>.3542</td>
<td>no</td>
</tr>
</tbody>
</table>

### Phase II

<table>
<thead>
<tr>
<th>Condition Pair</th>
<th>Probability</th>
<th>Significant?</th>
<th>Condition Pair</th>
<th>Probability</th>
<th>Significant?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA-MCS</td>
<td>.0001</td>
<td>yes</td>
<td>PSC-MARK</td>
<td>.0010</td>
<td>yes</td>
</tr>
<tr>
<td>SA-PSC</td>
<td>.0005</td>
<td>yes</td>
<td>PSC-INS</td>
<td>.0003</td>
<td>yes</td>
</tr>
<tr>
<td>SA-RD0</td>
<td>.0007</td>
<td>yes</td>
<td>PSC-REAL</td>
<td>.0013</td>
<td>yes</td>
</tr>
<tr>
<td>SA-RD1</td>
<td>.0065</td>
<td>yes</td>
<td>RD0-RD1</td>
<td>.0030</td>
<td>yes</td>
</tr>
<tr>
<td>SA-RD2</td>
<td>.0013</td>
<td>yes</td>
<td>RD0-RD2</td>
<td>.1125</td>
<td>no</td>
</tr>
<tr>
<td>SA-MARK</td>
<td>.0031</td>
<td>yes</td>
<td>RD0-MARK</td>
<td>.0016</td>
<td>yes</td>
</tr>
<tr>
<td>SA-INS</td>
<td>.0024</td>
<td>yes</td>
<td>RD0-INS</td>
<td>.0025</td>
<td>yes</td>
</tr>
<tr>
<td>SA-REAL</td>
<td>.0012</td>
<td>yes</td>
<td>RD0-REAL</td>
<td>.3717</td>
<td>no</td>
</tr>
<tr>
<td>MCS-PSC</td>
<td>.0135</td>
<td>yes</td>
<td>RD1-RD2</td>
<td>.0514</td>
<td>no</td>
</tr>
<tr>
<td>MCS-RD0</td>
<td>.0191</td>
<td>no</td>
<td>RD1-MARK</td>
<td>.0744</td>
<td>no</td>
</tr>
<tr>
<td>MCS-RD1</td>
<td>.0006</td>
<td>yes</td>
<td>RD1-INS</td>
<td>.1478</td>
<td>no</td>
</tr>
<tr>
<td>MCS-RD2</td>
<td>.0001</td>
<td>yes</td>
<td>RD1-REAL</td>
<td>.1074</td>
<td>no</td>
</tr>
<tr>
<td>MCS-MARK</td>
<td>.0001</td>
<td>yes</td>
<td>RD2-MARK</td>
<td>.0037</td>
<td>yes</td>
</tr>
<tr>
<td>MCS-INS</td>
<td>.0001</td>
<td>yes</td>
<td>RD2-INS</td>
<td>.0013</td>
<td>yes</td>
</tr>
<tr>
<td>MCS-REAL</td>
<td>.0018</td>
<td>yes</td>
<td>RD2-REAL</td>
<td>.5769</td>
<td>no</td>
</tr>
<tr>
<td>PSC-RD0</td>
<td>.3286</td>
<td>no</td>
<td>MARK-INS</td>
<td>.9024</td>
<td>no</td>
</tr>
<tr>
<td>PSC-RD1</td>
<td>.0088</td>
<td>yes</td>
<td>MARK-REAL</td>
<td>.0109</td>
<td>yes</td>
</tr>
<tr>
<td>PSC-RD2</td>
<td>.0025</td>
<td>yes</td>
<td>INS-REAL</td>
<td>.0051</td>
<td>yes</td>
</tr>
</tbody>
</table>
Appendix C

Raw Data

Subjects’ raw scores on all training and testing sentences are listed below. Conditions are listed in the order that they were presented. Although S4 received the same training in Phase I as S1–S3, no written responses were recorded.

Table C.1: Phase I Training Results.

<table>
<thead>
<tr>
<th>Condition</th>
<th>List</th>
<th>Total</th>
<th>S1 Score</th>
<th>S2 Score</th>
<th>S3 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>D</td>
<td>51</td>
<td>30</td>
<td>29</td>
<td>15</td>
</tr>
<tr>
<td>MCS</td>
<td>401-410</td>
<td>82</td>
<td>81</td>
<td>79</td>
<td>81</td>
</tr>
<tr>
<td>PSC</td>
<td>A</td>
<td>50</td>
<td>43</td>
<td>46</td>
<td>41</td>
</tr>
<tr>
<td>PSC</td>
<td>1-20</td>
<td>156</td>
<td>147</td>
<td>153</td>
<td>139</td>
</tr>
<tr>
<td>D1</td>
<td>F</td>
<td>50</td>
<td>43</td>
<td>39</td>
<td>30</td>
</tr>
<tr>
<td>SD2</td>
<td>41-50</td>
<td>79</td>
<td>69</td>
<td>73</td>
<td>71</td>
</tr>
<tr>
<td>D3</td>
<td>G</td>
<td>51</td>
<td>39</td>
<td>39</td>
<td>32</td>
</tr>
<tr>
<td>SD1</td>
<td>21-30</td>
<td>74</td>
<td>53</td>
<td>67</td>
<td>58</td>
</tr>
<tr>
<td>D5</td>
<td>H</td>
<td>50</td>
<td>25</td>
<td>42</td>
<td>31</td>
</tr>
<tr>
<td>D5</td>
<td>81-90</td>
<td>79</td>
<td>65</td>
<td>60</td>
<td>56</td>
</tr>
<tr>
<td>SD1</td>
<td>C</td>
<td>49</td>
<td>45</td>
<td>48</td>
<td>33</td>
</tr>
<tr>
<td>D3</td>
<td>71-80</td>
<td>80</td>
<td>67</td>
<td>73</td>
<td>64</td>
</tr>
<tr>
<td>SD2</td>
<td>E</td>
<td>51</td>
<td>49</td>
<td>50</td>
<td>42</td>
</tr>
<tr>
<td>D1</td>
<td>61-70</td>
<td>77</td>
<td>65</td>
<td>61</td>
<td>56</td>
</tr>
</tbody>
</table>

Letters in the List column represent CID lists and numbers represent Clarke sentences. Numbers in the Total column are the number of (key) words in a list. All subjects received the same training lists. Scores are number of (key) words correct.
Table C.2: Phase I Testing Results.

<table>
<thead>
<tr>
<th>Condition</th>
<th>S1 List</th>
<th>Score</th>
<th>S2 List</th>
<th>Score</th>
<th>S3 List</th>
<th>Score</th>
<th>S4 List</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>31</td>
<td>13</td>
<td>31</td>
<td>18</td>
<td>31</td>
<td>7</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>SA</td>
<td>35</td>
<td>13</td>
<td>35</td>
<td>10</td>
<td>35</td>
<td>12</td>
<td>18</td>
<td>11</td>
</tr>
<tr>
<td>SA</td>
<td>1</td>
<td>11</td>
<td>1</td>
<td>14</td>
<td>1</td>
<td>5</td>
<td>35</td>
<td>2</td>
</tr>
<tr>
<td>SA</td>
<td>18</td>
<td>23</td>
<td>18</td>
<td>23</td>
<td>18</td>
<td>23</td>
<td>31</td>
<td>6</td>
</tr>
<tr>
<td>MCS</td>
<td>40</td>
<td>39</td>
<td>40</td>
<td>42</td>
<td>40</td>
<td>39</td>
<td>40</td>
<td>32</td>
</tr>
<tr>
<td>MCS</td>
<td>41</td>
<td>45</td>
<td>41</td>
<td>46</td>
<td>41</td>
<td>40</td>
<td>41</td>
<td>29</td>
</tr>
<tr>
<td>MCS</td>
<td>42</td>
<td>43</td>
<td>42</td>
<td>47</td>
<td>42</td>
<td>35</td>
<td>42</td>
<td>16</td>
</tr>
<tr>
<td>MCS</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>46</td>
<td>43</td>
<td>44</td>
<td>43</td>
<td>26</td>
</tr>
<tr>
<td>PSC</td>
<td>29</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>57</td>
<td>36</td>
<td>2</td>
<td>34</td>
</tr>
<tr>
<td>PSC</td>
<td>38</td>
<td>43</td>
<td>17</td>
<td>41</td>
<td>9</td>
<td>44</td>
<td>3</td>
<td>26</td>
</tr>
<tr>
<td>PSC</td>
<td>58</td>
<td>42</td>
<td>14</td>
<td>40</td>
<td>65</td>
<td>31</td>
<td>4</td>
<td>33</td>
</tr>
<tr>
<td>PSC</td>
<td>60</td>
<td>43</td>
<td>47</td>
<td>43</td>
<td>15</td>
<td>38</td>
<td>6</td>
<td>34</td>
</tr>
<tr>
<td>D1</td>
<td>4</td>
<td>28</td>
<td>60</td>
<td>27</td>
<td>30</td>
<td>19</td>
<td>36</td>
<td>15</td>
</tr>
<tr>
<td>D1</td>
<td>17</td>
<td>23</td>
<td>58</td>
<td>33</td>
<td>14</td>
<td>20</td>
<td>37</td>
<td>17</td>
</tr>
<tr>
<td>SD2</td>
<td>2</td>
<td>42</td>
<td>57</td>
<td>36</td>
<td>29</td>
<td>29</td>
<td>60</td>
<td>19</td>
</tr>
<tr>
<td>SD2</td>
<td>27</td>
<td>27</td>
<td>9</td>
<td>40</td>
<td>16</td>
<td>24</td>
<td>65</td>
<td>17</td>
</tr>
<tr>
<td>D3</td>
<td>6</td>
<td>21</td>
<td>15</td>
<td>13</td>
<td>27</td>
<td>10</td>
<td>9</td>
<td>17</td>
</tr>
<tr>
<td>D3</td>
<td>14</td>
<td>28</td>
<td>16</td>
<td>20</td>
<td>60</td>
<td>18</td>
<td>56</td>
<td>8</td>
</tr>
<tr>
<td>SD1</td>
<td>3</td>
<td>28</td>
<td>28</td>
<td>29</td>
<td>66</td>
<td>16</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>SD1</td>
<td>37</td>
<td>27</td>
<td>29</td>
<td>27</td>
<td>36</td>
<td>29</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>D5</td>
<td>15</td>
<td>19</td>
<td>38</td>
<td>19</td>
<td>4</td>
<td>21</td>
<td>27</td>
<td>12</td>
</tr>
<tr>
<td>D5</td>
<td>16</td>
<td>18</td>
<td>66</td>
<td>18</td>
<td>68</td>
<td>18</td>
<td>28</td>
<td>12</td>
</tr>
<tr>
<td>D5</td>
<td>36</td>
<td>22</td>
<td>2</td>
<td>27</td>
<td>6</td>
<td>12</td>
<td>29</td>
<td>10</td>
</tr>
<tr>
<td>D5</td>
<td>56</td>
<td>22</td>
<td>3</td>
<td>24</td>
<td>47</td>
<td>11</td>
<td>30</td>
<td>14</td>
</tr>
<tr>
<td>SD1</td>
<td>47</td>
<td>35</td>
<td>4</td>
<td>33</td>
<td>56</td>
<td>19</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>SD1</td>
<td>57</td>
<td>36</td>
<td>65</td>
<td>26</td>
<td>38</td>
<td>23</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>D3</td>
<td>65</td>
<td>19</td>
<td>36</td>
<td>26</td>
<td>3</td>
<td>25</td>
<td>57</td>
<td>14</td>
</tr>
<tr>
<td>D3</td>
<td>68</td>
<td>24</td>
<td>37</td>
<td>20</td>
<td>17</td>
<td>20</td>
<td>58</td>
<td>17</td>
</tr>
<tr>
<td>SD2</td>
<td>28</td>
<td>36</td>
<td>6</td>
<td>34</td>
<td>58</td>
<td>26</td>
<td>66</td>
<td>21</td>
</tr>
<tr>
<td>SD2</td>
<td>30</td>
<td>35</td>
<td>56</td>
<td>34</td>
<td>37</td>
<td>32</td>
<td>68</td>
<td>20</td>
</tr>
<tr>
<td>D1</td>
<td>66</td>
<td>23</td>
<td>68</td>
<td>26</td>
<td>28</td>
<td>22</td>
<td>38</td>
<td>19</td>
</tr>
<tr>
<td>D1</td>
<td>9</td>
<td>34</td>
<td>27</td>
<td>24</td>
<td>2</td>
<td>34</td>
<td>47</td>
<td>30</td>
</tr>
<tr>
<td>SA</td>
<td>32</td>
<td>16</td>
<td>32</td>
<td>18</td>
<td>32</td>
<td>8</td>
<td>33</td>
<td>12</td>
</tr>
<tr>
<td>SA</td>
<td>33</td>
<td>18</td>
<td>33</td>
<td>14</td>
<td>33</td>
<td>5</td>
<td>32</td>
<td>6</td>
</tr>
<tr>
<td>MCS</td>
<td>48</td>
<td>44</td>
<td>48</td>
<td>45</td>
<td>48</td>
<td>40</td>
<td>48</td>
<td>25</td>
</tr>
<tr>
<td>MCS</td>
<td>49</td>
<td>47</td>
<td>49</td>
<td>48</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>34</td>
</tr>
</tbody>
</table>

All list numbers refer to Harvard lists. Scores are number of key words correct out of 50.
Letters in the List column represent CID lists and numbers represent Clarke sentences. Numbers in the Total column are the number of (key) words in a list. All subjects received the same training lists. Scores are number of (key) words correct.

<table>
<thead>
<tr>
<th>Condition</th>
<th>List</th>
<th>Total</th>
<th>S1 Score</th>
<th>S2 Score</th>
<th>S3 Score</th>
<th>S5 Score</th>
<th>S6 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>131–140</td>
<td>80</td>
<td>66</td>
<td>67</td>
<td>47</td>
<td>68</td>
<td>52</td>
</tr>
<tr>
<td>MCS</td>
<td>411–420</td>
<td>75</td>
<td>71</td>
<td>73</td>
<td>66</td>
<td>75</td>
<td>74</td>
</tr>
<tr>
<td>PSC</td>
<td>101–120</td>
<td>160</td>
<td>153</td>
<td>148</td>
<td>149</td>
<td>143</td>
<td>151</td>
</tr>
<tr>
<td>MARK</td>
<td>211–220</td>
<td>71</td>
<td>65</td>
<td>65</td>
<td>57</td>
<td>68</td>
<td>67</td>
</tr>
<tr>
<td>RD2</td>
<td>171–180</td>
<td>81</td>
<td>79</td>
<td>78</td>
<td>74</td>
<td>81</td>
<td>73</td>
</tr>
<tr>
<td>RD1</td>
<td>151–160</td>
<td>82</td>
<td>78</td>
<td>67</td>
<td>72</td>
<td>74</td>
<td>70</td>
</tr>
<tr>
<td>REAL</td>
<td>J</td>
<td>51</td>
<td>41</td>
<td>46</td>
<td>44</td>
<td>46</td>
<td>42</td>
</tr>
<tr>
<td>RD0</td>
<td>121–130</td>
<td>78</td>
<td>75</td>
<td>72</td>
<td>70</td>
<td>76</td>
<td>72</td>
</tr>
<tr>
<td>INS</td>
<td>191–200</td>
<td>74</td>
<td>57</td>
<td>66</td>
<td>62</td>
<td>72</td>
<td>70</td>
</tr>
<tr>
<td>MARK</td>
<td>221–230</td>
<td>80</td>
<td>76</td>
<td>69</td>
<td>75</td>
<td>68</td>
<td>79</td>
</tr>
<tr>
<td>RD0</td>
<td>141–150</td>
<td>80</td>
<td>72</td>
<td>74</td>
<td>78</td>
<td>73</td>
<td>75</td>
</tr>
<tr>
<td>RD2</td>
<td>181–190</td>
<td>85</td>
<td>80</td>
<td>73</td>
<td>73</td>
<td>80</td>
<td>83</td>
</tr>
<tr>
<td>INS</td>
<td>201–210</td>
<td>80</td>
<td>74</td>
<td>74</td>
<td>76</td>
<td>73</td>
<td>66</td>
</tr>
<tr>
<td>REAL</td>
<td>91–100</td>
<td>79</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>76</td>
</tr>
<tr>
<td>RD1</td>
<td>161–170</td>
<td>80</td>
<td>72</td>
<td>74</td>
<td>71</td>
<td>80</td>
<td>67</td>
</tr>
</tbody>
</table>
Table C.4: Phase II Testing Results.

<table>
<thead>
<tr>
<th>Condition</th>
<th>S1 List</th>
<th>S1 Score</th>
<th>S2 List</th>
<th>S2 Score</th>
<th>S3 List</th>
<th>S3 Score</th>
<th>S4 List</th>
<th>S4 Score</th>
<th>S5 List</th>
<th>S5 Score</th>
<th>S6 List</th>
<th>S6 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>46</td>
<td>24</td>
<td>5</td>
<td>19</td>
<td>5</td>
<td>15</td>
<td>5</td>
<td>19</td>
<td>5</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SA</td>
<td>34</td>
<td>15</td>
<td>8</td>
<td>15</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>13</td>
<td>8</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCS</td>
<td>5</td>
<td>48</td>
<td>34</td>
<td>46</td>
<td>61</td>
<td>41</td>
<td>61</td>
<td>33</td>
<td>34</td>
<td>47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCS</td>
<td>8</td>
<td>43</td>
<td>24</td>
<td>48</td>
<td>67</td>
<td>48</td>
<td>67</td>
<td>40</td>
<td>24</td>
<td>48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSC</td>
<td>50</td>
<td>41</td>
<td>26</td>
<td>36</td>
<td>51</td>
<td>41</td>
<td>20</td>
<td>35</td>
<td>26</td>
<td>41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSC</td>
<td>52</td>
<td>44</td>
<td>45</td>
<td>44</td>
<td>55</td>
<td>43</td>
<td>21</td>
<td>37</td>
<td>45</td>
<td>37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARK</td>
<td>26</td>
<td>26</td>
<td>22</td>
<td>31</td>
<td>45</td>
<td>31</td>
<td>55</td>
<td>29</td>
<td>22</td>
<td>33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARK</td>
<td>70</td>
<td>34</td>
<td>23</td>
<td>29</td>
<td>54</td>
<td>-</td>
<td>63</td>
<td>20</td>
<td>23</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD2</td>
<td>22</td>
<td>45</td>
<td>19</td>
<td>36</td>
<td>13</td>
<td>31</td>
<td>64</td>
<td>28</td>
<td>19</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD2</td>
<td>24</td>
<td>30</td>
<td>21</td>
<td>38</td>
<td>59</td>
<td>40</td>
<td>70</td>
<td>29</td>
<td>21</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD1</td>
<td>71</td>
<td>33</td>
<td>72</td>
<td>37</td>
<td>22</td>
<td>33</td>
<td>52</td>
<td>26</td>
<td>72</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD1</td>
<td>7</td>
<td>29</td>
<td>55</td>
<td>37</td>
<td>24</td>
<td>25</td>
<td>54</td>
<td>14</td>
<td>55</td>
<td>34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td>13</td>
<td>37</td>
<td>50</td>
<td>35</td>
<td>69</td>
<td>20</td>
<td>7</td>
<td>31</td>
<td>50</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td>19</td>
<td>39</td>
<td>51</td>
<td>39</td>
<td>70</td>
<td>39</td>
<td>10</td>
<td>35</td>
<td>51</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD0</td>
<td>23</td>
<td>36</td>
<td>59</td>
<td>47</td>
<td>19</td>
<td>34</td>
<td>24</td>
<td>33</td>
<td>59</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD0</td>
<td>53</td>
<td>34</td>
<td>70</td>
<td>40</td>
<td>20</td>
<td>40</td>
<td>26</td>
<td>35</td>
<td>70</td>
<td>48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INS</td>
<td>45</td>
<td>26</td>
<td>13</td>
<td>20</td>
<td>23</td>
<td>25</td>
<td>44</td>
<td>14</td>
<td>13</td>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INS</td>
<td>64</td>
<td>23</td>
<td>20</td>
<td>41</td>
<td>63</td>
<td>30</td>
<td>46</td>
<td>30</td>
<td>20</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARK</td>
<td>44</td>
<td>27</td>
<td>10</td>
<td>29</td>
<td>64</td>
<td>20</td>
<td>69</td>
<td>12</td>
<td>10</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MARK</td>
<td>51</td>
<td>36</td>
<td>11</td>
<td>34</td>
<td>53</td>
<td>22</td>
<td>71</td>
<td>32</td>
<td>11</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD0</td>
<td>63</td>
<td>36</td>
<td>12</td>
<td>42</td>
<td>21</td>
<td>34</td>
<td>45</td>
<td>34</td>
<td>12</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD0</td>
<td>69</td>
<td>42</td>
<td>7</td>
<td>45</td>
<td>52</td>
<td>23</td>
<td>50</td>
<td>32</td>
<td>7</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD2</td>
<td>54</td>
<td>30</td>
<td>69</td>
<td>40</td>
<td>44</td>
<td>33</td>
<td>72</td>
<td>36</td>
<td>69</td>
<td>33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD2</td>
<td>62</td>
<td>37</td>
<td>71</td>
<td>43</td>
<td>46</td>
<td>30</td>
<td>25</td>
<td>28</td>
<td>71</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INS</td>
<td>55</td>
<td>36</td>
<td>54</td>
<td>33</td>
<td>10</td>
<td>31</td>
<td>51</td>
<td>33</td>
<td>54</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INS</td>
<td>11</td>
<td>35</td>
<td>62</td>
<td>28</td>
<td>12</td>
<td>34</td>
<td>53</td>
<td>21</td>
<td>62</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td>20</td>
<td>41</td>
<td>52</td>
<td>37</td>
<td>71</td>
<td>40</td>
<td>11</td>
<td>35</td>
<td>52</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td>21</td>
<td>42</td>
<td>53</td>
<td>35</td>
<td>72</td>
<td>42</td>
<td>12</td>
<td>30</td>
<td>53</td>
<td>34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD1</td>
<td>10</td>
<td>35</td>
<td>63</td>
<td>31</td>
<td>26</td>
<td>19</td>
<td>59</td>
<td>26</td>
<td>63</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD1</td>
<td>12</td>
<td>34</td>
<td>25</td>
<td>37</td>
<td>50</td>
<td>29</td>
<td>62</td>
<td>23</td>
<td>25</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSC</td>
<td>59</td>
<td>43</td>
<td>64</td>
<td>46</td>
<td>7</td>
<td>44</td>
<td>22</td>
<td>44</td>
<td>64</td>
<td>34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSC</td>
<td>72</td>
<td>43</td>
<td>44</td>
<td>44</td>
<td>11</td>
<td>39</td>
<td>23</td>
<td>37</td>
<td>44</td>
<td>45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCS</td>
<td>25</td>
<td>48</td>
<td>46</td>
<td>46</td>
<td>34</td>
<td>41</td>
<td>13</td>
<td>44</td>
<td>46</td>
<td>42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCS</td>
<td>39</td>
<td>45</td>
<td>39</td>
<td>47</td>
<td>39</td>
<td>43</td>
<td>19</td>
<td>46</td>
<td>39</td>
<td>46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SA</td>
<td>61</td>
<td>23</td>
<td>61</td>
<td>25</td>
<td>62</td>
<td>8</td>
<td>34</td>
<td>16</td>
<td>61</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SA</td>
<td>67</td>
<td>25</td>
<td>67</td>
<td>22</td>
<td>25</td>
<td>10</td>
<td>39</td>
<td>27</td>
<td>67</td>
<td>25</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All list numbers refer to Harvard lists. Scores are number of key words correct out of 50. S1 received the last 2 SA lists before the last 2 MCS lists. Score for S2 and S6 on RD2 list 43 are out of 45 key words. S3 did not receive MARK list 54.
Appendix D

Transcription Files and Playlists

Examples of a transcription file, the *.lgp output of program S2 and a playlist for Harvard sentence ‘The bunch of grapes was pressed into wine’ (4703) are provided here and discussed in Sections 2.2 and 2.4. The position coordinates for the CID, Clarke, and Harvard sentences are listed at the end of the appendix.
Transcription file from acoustic waveform

Start times (in msec) for each phone begin at the onset of the acoustic signal. The backslash is used to separate the times from the syllable symbols and phones.

1401.3/‘dh
1425.9/,ax
1479.9/‘bcl
1593.6/+b
1602.8/,ah
1696.1/*n
1770.5/*ch
1842.2/‘ax
1897.1/*v
1941.7/.gcl
2015.6/+g
2072.8/‘r
2140.8/,ey
2274.1/*pcl
2338.1/+p
2351.3/*s
2484.6/‘w
2505.3/,ax
2551.3/*z
2621.3/.pcl
2707.7/+p
2794.6/‘r
2830.1/,eh
2895.8/*s
2992.3/*tcl
3027.2/+t
3058.5/‘ih
3099.6/*n
3149.8/‘tcl
3185.1/+t
3241.5/,uw
3387.3/‘w
3504/,ay
3763.1/*n
3919.8/¨
Output of $S2$ indicating errors

Phones on the right come from the acoustic transcriptions. Phones with stars to their right received errors. The replacement phones are listed on the left and are used by State to create the playlists.

1401.3/‘xxx
dh,*
1425.9/,ax
ax ,
1479.9/‘bcl
bcl ,
1593.6/+t
b ,*
1602.8/,ah
ah ,
1696.1/*n
n ,
1770.5/*ch
ch ,
1842.2/‘ow
ax ,*
1897.1/*v
v ,
1941.7/.gcl
gcl ,
2015.6/+g
g ,
2072.8/‘r
r ,
2140.8/,ey
ey ,
2274.1/*pcl
pcl ,
2338.1/+p
p ,
2351.3/*s
s ,
2484.6/‘w
w ,
2505.3/,er
ax ,*
2551.3/*s
z ,*
2621.3/.pcl
pcl ,
2707.7/+ow
p ,*
2794.6/‘r
r ,
2830.1/,eh
eh ,
2895.8/*s
s ,
2992.3/*tcl
tcl ,
3027.2/+t
t ,
3058.5/‘ih
ih ,
3099.6/*n
n ,
3149.8/‘tcl
tcl ,
3185.1/+t
t ,
3241.5/,uw
uw ,
3387.3/‘w
w ,
3504/,ay
ay ,
3763.1/*m
n ,*
3919.8/\~
Playlist (output of State)

*Positions:* position code, x-y screen coordinates.


*Sentences:* sentence code, sentence start frame, sentence stop frame.

*Playlists:* sentence code, cue start frame, cue stop frame, position code, handshape code. # signifies a comment.

```
POsITIONS
{
  # d side
  0 90 300
  # g side
  1 105 300
  # h side
  2 105 300
  # k side
  3 105 300
  # l side
  4 105 300
  # n side
  5 100 300
  # ng side
  6 115 300
  # t side
  7 105 300
  # d mouth
  8 145 300
  # g mouth
  9 165 300
  # h mouth
  10 185 300
  # k mouth
  11 185 300
  # l mouth
  12 175 300
  # n mouth
  13 170 300
  # ng mouth
  14 170 270
  # t mouth
  15 145 300
  # d throat
  16 183 395
  # g throat
```
17 205 395
# h throat
18 213 395
# k throat
19 225 395
# l throat
20 220 395
# n throat
21 205 395
# ng throat
22 220 395
# t throat
23 180 395
# d chin
24 175 350
# g chin
25 200 350
# h chin
26 215 350
# k chin
27 225 350
# l chin
28 215 350
# n chin
29 200 350
# ng chin
30 235 350
# t chin
31 185 350
# d side-down
32 90 350
# g side-down
33 105 350
# h,k,l side-down
34 105 350
# k side-down
35 105 350
# l side-down
36 105 350
# n side-down
37 105 350
# ng side-down
38 105 350
# t side-down
39 105 350
Table D.1: CID sentences position coordinates

<table>
<thead>
<tr>
<th>Cue Group</th>
<th>Position</th>
<th>side</th>
<th>mouth</th>
<th>throat</th>
<th>chin</th>
<th>side-down</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>90 200</td>
<td>215 210</td>
<td>250 295</td>
<td>245 250</td>
<td>110 250</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>105 200</td>
<td>245 210</td>
<td>290 295</td>
<td>270 250</td>
<td>135 250</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>105 200</td>
<td>255 210</td>
<td>290 295</td>
<td>295 250</td>
<td>135 250</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>105 200</td>
<td>255 210</td>
<td>300 295</td>
<td>295 250</td>
<td>135 250</td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>105 200</td>
<td>265 210</td>
<td>300 295</td>
<td>285 250</td>
<td>135 250</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>100 200</td>
<td>240 210</td>
<td>290 295</td>
<td>270 250</td>
<td>135 250</td>
<td></td>
</tr>
<tr>
<td>ng</td>
<td>115 200</td>
<td>240 180</td>
<td>295 295</td>
<td>315 250</td>
<td>135 250</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>105 200</td>
<td>225 195</td>
<td>270 295</td>
<td>245 240</td>
<td>125 250</td>
<td></td>
</tr>
</tbody>
</table>

Table D.2: Clarke sentences position coordinates

<table>
<thead>
<tr>
<th>Cue Group</th>
<th>Position</th>
<th>side</th>
<th>mouth</th>
<th>throat</th>
<th>chin</th>
<th>side-down</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>90 250</td>
<td>168 225</td>
<td>200 315</td>
<td>205 265</td>
<td>90 285</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>120 250</td>
<td>182 225</td>
<td>220 320</td>
<td>225 255</td>
<td>120 285</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>135 250</td>
<td>205 220</td>
<td>235 320</td>
<td>245 260</td>
<td>135 285</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>135 250</td>
<td>205 225</td>
<td>235 320</td>
<td>245 260</td>
<td>135 285</td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>135 250</td>
<td>205 225</td>
<td>235 320</td>
<td>245 260</td>
<td>135 285</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>110 240</td>
<td>180 210</td>
<td>235 320</td>
<td>220 255</td>
<td>110 275</td>
<td></td>
</tr>
<tr>
<td>ng</td>
<td>120 250</td>
<td>182 190</td>
<td>255 320</td>
<td>275 265</td>
<td>120 285</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>90 250</td>
<td>155 210</td>
<td>210 320</td>
<td>195 250</td>
<td>90 285</td>
<td></td>
</tr>
</tbody>
</table>

Table D.3: Harvard sentences (1–58) position coordinates

<table>
<thead>
<tr>
<th>Cue Group</th>
<th>Position</th>
<th>side</th>
<th>mouth</th>
<th>throat</th>
<th>chin</th>
<th>side-down</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>90 300</td>
<td>145 300</td>
<td>183 395</td>
<td>175 350</td>
<td>90 350</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>105 300</td>
<td>165 300</td>
<td>205 395</td>
<td>200 350</td>
<td>105 350</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>105 300</td>
<td>185 300</td>
<td>213 395</td>
<td>215 350</td>
<td>105 350</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>105 300</td>
<td>185 300</td>
<td>225 395</td>
<td>225 350</td>
<td>105 350</td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>105 300</td>
<td>175 300</td>
<td>220 395</td>
<td>215 350</td>
<td>105 350</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>100 300</td>
<td>170 300</td>
<td>205 395</td>
<td>200 350</td>
<td>105 350</td>
<td></td>
</tr>
<tr>
<td>ng</td>
<td>115 300</td>
<td>170 270</td>
<td>220 395</td>
<td>235 350</td>
<td>105 350</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>105 300</td>
<td>145 300</td>
<td>180 395</td>
<td>185 350</td>
<td>105 350</td>
<td></td>
</tr>
</tbody>
</table>
Table D.4: Harvard sentences (59–72) position coordinates

<table>
<thead>
<tr>
<th>Cue Group</th>
<th>side</th>
<th>mouth</th>
<th>throat</th>
<th>chin</th>
<th>side-down</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>80 300</td>
<td>130 290</td>
<td>183 395</td>
<td>175 340</td>
<td>80 350</td>
</tr>
<tr>
<td>g</td>
<td>95 300</td>
<td>150 290</td>
<td>205 395</td>
<td>200 340</td>
<td>95 350</td>
</tr>
<tr>
<td>h</td>
<td>105 300</td>
<td>170 285</td>
<td>213 395</td>
<td>215 340</td>
<td>105 350</td>
</tr>
<tr>
<td>k</td>
<td>105 300</td>
<td>170 290</td>
<td>225 395</td>
<td>225 340</td>
<td>105 350</td>
</tr>
<tr>
<td>l</td>
<td>105 300</td>
<td>165 290</td>
<td>220 395</td>
<td>215 340</td>
<td>105 350</td>
</tr>
<tr>
<td>n</td>
<td>90 300</td>
<td>150 280</td>
<td>205 395</td>
<td>200 335</td>
<td>90 350</td>
</tr>
<tr>
<td>ng</td>
<td>105 290</td>
<td>155 260</td>
<td>220 395</td>
<td>235 345</td>
<td>105 350</td>
</tr>
<tr>
<td>t</td>
<td>85 290</td>
<td>130 270</td>
<td>180 395</td>
<td>185 330</td>
<td>85 350</td>
</tr>
</tbody>
</table>
Appendix E

Confusion Matrices

The following confusion matrices are adaptations of confusion matrix D.1. from Paul Duchnowski's dissertation [16]. Input phones are listed along the left most column and output phones are listed across the top row. The matrix entries are percents of input phones identified as output phones times ten, allowing the matrices to fit on one page while preserving the decimal place. Percents less than one are not listed. The program S2 accepted matrices with entries in number of phones rather than percentages. Table E.3 lists the total number of input phones for each row of CM1 and CM2, which were used to convert between percentages and number of phones. S2 received CM1 and CM2 with their entries multiplied by ten to preserve the decimal place.
Table E.1: Confusion matrix CM\textsubscript{1} used to generate errors in conditions SD1, SD2, D1, D3, D5, RD1, RD2, MARK.

<table>
<thead>
<tr>
<th>Input Phone</th>
<th>Output Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>iy</td>
<td>iy</td>
</tr>
<tr>
<td>th</td>
<td>th</td>
</tr>
<tr>
<td>nh</td>
<td>nh</td>
</tr>
<tr>
<td>sh</td>
<td>sh</td>
</tr>
<tr>
<td>eo</td>
<td>eo</td>
</tr>
<tr>
<td>ut</td>
<td>ut</td>
</tr>
<tr>
<td>aw</td>
<td>aw</td>
</tr>
<tr>
<td>ow</td>
<td>ow</td>
</tr>
<tr>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>er</td>
<td>er</td>
</tr>
<tr>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>en</td>
<td>en</td>
</tr>
<tr>
<td>ng</td>
<td>ng</td>
</tr>
<tr>
<td>ch</td>
<td>ch</td>
</tr>
<tr>
<td>jh</td>
<td>jh</td>
</tr>
<tr>
<td>dh</td>
<td>dh</td>
</tr>
<tr>
<td>dx</td>
<td>dx</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
</tr>
<tr>
<td>p</td>
<td>p</td>
</tr>
<tr>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>sh</td>
<td>sh</td>
</tr>
<tr>
<td>v</td>
<td>v</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>th</td>
<td>th</td>
</tr>
<tr>
<td>sh</td>
<td>sh</td>
</tr>
<tr>
<td>h</td>
<td>h</td>
</tr>
<tr>
<td>bcl</td>
<td>bcl</td>
</tr>
<tr>
<td>del</td>
<td>del</td>
</tr>
<tr>
<td>gel</td>
<td>gel</td>
</tr>
<tr>
<td>kcl</td>
<td>kcl</td>
</tr>
<tr>
<td>pcl</td>
<td>pcl</td>
</tr>
<tr>
<td>tel</td>
<td>tel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input Phone</th>
<th>Output Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>iy</td>
<td>iy</td>
</tr>
<tr>
<td>th</td>
<td>th</td>
</tr>
<tr>
<td>nh</td>
<td>nh</td>
</tr>
<tr>
<td>sh</td>
<td>sh</td>
</tr>
<tr>
<td>eo</td>
<td>eo</td>
</tr>
<tr>
<td>ut</td>
<td>ut</td>
</tr>
<tr>
<td>aw</td>
<td>aw</td>
</tr>
<tr>
<td>ow</td>
<td>ow</td>
</tr>
<tr>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>er</td>
<td>er</td>
</tr>
<tr>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>en</td>
<td>en</td>
</tr>
<tr>
<td>ng</td>
<td>ng</td>
</tr>
<tr>
<td>ch</td>
<td>ch</td>
</tr>
<tr>
<td>jh</td>
<td>jh</td>
</tr>
<tr>
<td>dh</td>
<td>dh</td>
</tr>
<tr>
<td>dx</td>
<td>dx</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
</tr>
<tr>
<td>p</td>
<td>p</td>
</tr>
<tr>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>sh</td>
<td>sh</td>
</tr>
<tr>
<td>v</td>
<td>v</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>th</td>
<td>th</td>
</tr>
<tr>
<td>sh</td>
<td>sh</td>
</tr>
<tr>
<td>h</td>
<td>h</td>
</tr>
<tr>
<td>bcl</td>
<td>bcl</td>
</tr>
<tr>
<td>del</td>
<td>del</td>
</tr>
<tr>
<td>gel</td>
<td>gel</td>
</tr>
<tr>
<td>kcl</td>
<td>kcl</td>
</tr>
<tr>
<td>pcl</td>
<td>pcl</td>
</tr>
<tr>
<td>tel</td>
<td>tel</td>
</tr>
</tbody>
</table>
Table E.2: Confusion matrix CM2 used to generate errors in condition INS.

<table>
<thead>
<tr>
<th>Input Phone</th>
<th>Output Phone</th>
<th>Input Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>iy</td>
<td>nh</td>
<td>iy</td>
</tr>
<tr>
<td>ih</td>
<td>nh</td>
<td>ih</td>
</tr>
<tr>
<td>ee</td>
<td>nh</td>
<td>ee</td>
</tr>
<tr>
<td>ea</td>
<td>nh</td>
<td>ea</td>
</tr>
<tr>
<td>ax</td>
<td>nh</td>
<td>ax</td>
</tr>
<tr>
<td>ah</td>
<td>nh</td>
<td>ah</td>
</tr>
<tr>
<td>uh</td>
<td>nh</td>
<td>uh</td>
</tr>
<tr>
<td>ao</td>
<td>nh</td>
<td>ao</td>
</tr>
<tr>
<td>ay</td>
<td>nh</td>
<td>ay</td>
</tr>
<tr>
<td>ea</td>
<td>nh</td>
<td>ea</td>
</tr>
<tr>
<td>aw</td>
<td>nh</td>
<td>aw</td>
</tr>
<tr>
<td>ow</td>
<td>nh</td>
<td>ow</td>
</tr>
<tr>
<td>t</td>
<td>nh</td>
<td>t</td>
</tr>
<tr>
<td>r</td>
<td>nh</td>
<td>r</td>
</tr>
<tr>
<td>y</td>
<td>nh</td>
<td>y</td>
</tr>
<tr>
<td>w</td>
<td>nh</td>
<td>w</td>
</tr>
<tr>
<td>er</td>
<td>nh</td>
<td>er</td>
</tr>
<tr>
<td>m</td>
<td>nh</td>
<td>m</td>
</tr>
<tr>
<td>n</td>
<td>nh</td>
<td>n</td>
</tr>
<tr>
<td>ng</td>
<td>nh</td>
<td>ng</td>
</tr>
<tr>
<td>ch</td>
<td>nh</td>
<td>ch</td>
</tr>
<tr>
<td>jh</td>
<td>nh</td>
<td>jh</td>
</tr>
<tr>
<td>dx</td>
<td>nh</td>
<td>dx</td>
</tr>
<tr>
<td>b</td>
<td>nh</td>
<td>b</td>
</tr>
<tr>
<td>d</td>
<td>nh</td>
<td>d</td>
</tr>
<tr>
<td>g</td>
<td>nh</td>
<td>g</td>
</tr>
<tr>
<td>p</td>
<td>nh</td>
<td>p</td>
</tr>
<tr>
<td>t</td>
<td>nh</td>
<td>t</td>
</tr>
<tr>
<td>k</td>
<td>nh</td>
<td>k</td>
</tr>
<tr>
<td>sh</td>
<td>nh</td>
<td>sh</td>
</tr>
<tr>
<td>v</td>
<td>nh</td>
<td>v</td>
</tr>
<tr>
<td>f</td>
<td>nh</td>
<td>f</td>
</tr>
<tr>
<td>th</td>
<td>nh</td>
<td>th</td>
</tr>
<tr>
<td>s</td>
<td>nh</td>
<td>s</td>
</tr>
<tr>
<td>h</td>
<td>nh</td>
<td>h</td>
</tr>
<tr>
<td>bcl</td>
<td>nh</td>
<td>bcl</td>
</tr>
<tr>
<td>dcl</td>
<td>nh</td>
<td>dcl</td>
</tr>
<tr>
<td>gcl</td>
<td>nh</td>
<td>gcl</td>
</tr>
<tr>
<td>kcl</td>
<td>nh</td>
<td>kcl</td>
</tr>
<tr>
<td>pcl</td>
<td>nh</td>
<td>pcl</td>
</tr>
<tr>
<td>tcl</td>
<td>nh</td>
<td>tcl</td>
</tr>
</tbody>
</table>

78
Table E.3: Number of input phones in the matrices CM1 and CM2.

<table>
<thead>
<tr>
<th>Input Phone</th>
<th>CM1</th>
<th>CM2</th>
<th>Cue Group</th>
<th>Input Phone</th>
<th>CM1</th>
<th>CM2</th>
<th>Cue Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>iy</td>
<td>2406</td>
<td>2410</td>
<td>mouth</td>
<td>ch</td>
<td>184</td>
<td>207</td>
<td>ng-handshape</td>
</tr>
<tr>
<td>ih</td>
<td>2179</td>
<td>2085</td>
<td>throat</td>
<td>jh</td>
<td>213</td>
<td>233</td>
<td>g-handshape</td>
</tr>
<tr>
<td>eh</td>
<td>669</td>
<td>657</td>
<td>chin</td>
<td>dh</td>
<td>544</td>
<td>519</td>
<td>k-handshape</td>
</tr>
<tr>
<td>ae</td>
<td>523</td>
<td>526</td>
<td>throat</td>
<td>dx</td>
<td>336</td>
<td>311</td>
<td>d-handshape</td>
</tr>
<tr>
<td>ax</td>
<td>797</td>
<td>752</td>
<td>side-down</td>
<td>b</td>
<td>571</td>
<td>572</td>
<td>n-handshape</td>
</tr>
<tr>
<td>ah</td>
<td>440</td>
<td>446</td>
<td>side-down</td>
<td>d</td>
<td>453</td>
<td>450</td>
<td>d-handshape</td>
</tr>
<tr>
<td>uw</td>
<td>339</td>
<td>351</td>
<td>chin</td>
<td>g</td>
<td>311</td>
<td>323</td>
<td>g-handshape</td>
</tr>
<tr>
<td>uh</td>
<td>130</td>
<td>147</td>
<td>throat</td>
<td>p</td>
<td>588</td>
<td>588</td>
<td>d-handshape</td>
</tr>
<tr>
<td>ao</td>
<td>501</td>
<td>503</td>
<td>chin</td>
<td>t</td>
<td>731</td>
<td>711</td>
<td>t-handshape</td>
</tr>
<tr>
<td>aa</td>
<td>562</td>
<td>566</td>
<td>side-front</td>
<td>k</td>
<td>754</td>
<td>746</td>
<td>k-handshape</td>
</tr>
<tr>
<td>ey</td>
<td>435</td>
<td>455</td>
<td>chin-throat</td>
<td>z</td>
<td>686</td>
<td>701</td>
<td>k-handshape</td>
</tr>
<tr>
<td>ay</td>
<td>422</td>
<td>438</td>
<td>side-throat</td>
<td>zh</td>
<td>57</td>
<td>78</td>
<td>d-handshape</td>
</tr>
<tr>
<td>oy</td>
<td>96</td>
<td>117</td>
<td>chin-throat</td>
<td>v</td>
<td>394</td>
<td>380</td>
<td>k-handshape</td>
</tr>
<tr>
<td>aw</td>
<td>124</td>
<td>145</td>
<td>side-throat</td>
<td>f</td>
<td>549</td>
<td>561</td>
<td>t-handshape</td>
</tr>
<tr>
<td>ow</td>
<td>353</td>
<td>354</td>
<td>side-front</td>
<td>th</td>
<td>157</td>
<td>174</td>
<td>g-handshape</td>
</tr>
<tr>
<td>l</td>
<td>1344</td>
<td>1271</td>
<td>l-handshape</td>
<td>s</td>
<td>1205</td>
<td>1218</td>
<td>h-handshape</td>
</tr>
<tr>
<td>r</td>
<td>1161</td>
<td>1089</td>
<td>h-handshape</td>
<td>sh</td>
<td>254</td>
<td>274</td>
<td>l-handshape</td>
</tr>
<tr>
<td>y</td>
<td>263</td>
<td>256</td>
<td>ng-handshape</td>
<td>h</td>
<td>251</td>
<td>258</td>
<td>h-handshape</td>
</tr>
<tr>
<td>w</td>
<td>546</td>
<td>542</td>
<td>l-handshape</td>
<td>bcl</td>
<td>1</td>
<td>23</td>
<td>n-handshape</td>
</tr>
<tr>
<td>er</td>
<td>1089</td>
<td>1086</td>
<td>mouth</td>
<td>dcl</td>
<td>1</td>
<td>21</td>
<td>d-handshape</td>
</tr>
<tr>
<td>m</td>
<td>835</td>
<td>831</td>
<td>t-handshape</td>
<td>gcl</td>
<td>1</td>
<td>24</td>
<td>g-handshape</td>
</tr>
<tr>
<td>n</td>
<td>1388</td>
<td>1330</td>
<td>n-handshape</td>
<td>kcl</td>
<td>1</td>
<td>25</td>
<td>k-handshape</td>
</tr>
<tr>
<td>en</td>
<td>115</td>
<td>136</td>
<td>chin/n-handshape</td>
<td>pcl</td>
<td>1</td>
<td>24</td>
<td>d-handshape</td>
</tr>
<tr>
<td>ng</td>
<td>207</td>
<td>222</td>
<td>ng-handshape</td>
<td>tcl</td>
<td>1</td>
<td>26</td>
<td>t-handshape</td>
</tr>
</tbody>
</table>
Appendix F

Source Code

The following code was used to convert phonetic transcription files into playlist files as discussed in Section 2.2. The State program is listed first in its' entirety, followed by various versions of State used to introduce delays, cue markings and insertion errors. Only the sections of State that were edited to create the other versions is included. Finally, a new version of State is included at the end which can convert output straight from the HTK recognizer into playlist files.

state.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXLEN 80 /* maximum number of lines in input file */
#define f .96432    /* correction factor for sampling */

void state1(FILE *, int *, char phon[MAXLEN][4], int, char sent[1][10], char *);
void state2(FILE *, int *, char phon[MAXLEN][4], int, char sent[1][10], char *);
void state2A(FILE *, int *, char phon[MAXLEN][4], int, char sent[1][10], char *);
void state3(FILE *, int *, char phon[MAXLEN][4], int, char sent[1][10], char *);
void state4(FILE *, int *, char phon[MAXLEN][4], int, int, char sent[1][10], char *);
void state5(FILE *, int *, char phon[MAXLEN][4], int, int, char sent[1][10], char *);
void timeout(int *, int *, int, int);
int frameout(double *, int *, int);

main(int argc, char *argv[])
```
{ FILE *sent_file; /* pointer to file to read in */
FILE *frame_file; /* pointer to file that has frame info */
FILE *out_file; /* pointer to file to put output */
char filename[30]; /* name of file to read in */
char phon[MAXLEN][4]; /* 2D array to hold phonemes lineXphon. length */
char sent[1][10]; /* array for sentence id */
int startfm, stopfm; /* beginning frame and end frame (nonblue) */
double ftm[MAXLEN]; /* array to hold time values */
char cue[MAXLEN]; /* array to hold cue markings */
int frame[MAXLEN]; /* array to hold absolute frames numbers */
int i = 0, j, k; /* array index */

/* Get name of file from user */
if (argc == 1)
    { printf("Input name of sentence file > ");
      scanf("%s", filename);
      sent_file = fopen(filename, "r");
      out_file = stdout;}
else if (argc == 2)
    { sent_file = fopen(argv[1], "r");
      out_file = stdout;}
else if (argc == 3)
    { sent_file = fopen(argv[1], "r");
      outfile = fopen(argv[2], "w");}

/* Divide file components (times, cues, phonemes) into separate arrays */
while (fscanf(sent_file, "%lf/%cYs", &ftm[i], &cue[i], phon[i]) != EOF)
    i++;
fclose(sent_file);

/* Reads file that contains information on sentence */
/* and start and stop frames for the sentence (nonblue) */
frame_file = fopen("finfo.dat", "r");
if (fscanf(frame_file, "%s%d%d", sent[0], &startfm, &stopfm) != EOF)
    .;
fclose(frame_file);

/* Convert times to frame values. Exit if overwriting into blue. */
if ((frameout(ftm, frame, startfm, stopfm)) == 0)
    printf("IN SENTENCE %s\n", sent[0]);

fprintf(out_file, "\nSENTENCES\n{\n");
fprintf(out_file, "%s %0.5d %0.5d\n", sent[0], startfm-2, stopfm+3);
fprintf(out_file, "\nPLAYLIST\n{\n");

/* Implement the state diagram */
for (j=0; cue[j] != \"\0\", j++)
    {switch(cue[j])
        {case '\n':
            break;
        case '\n':
            if (cue[j+1] == '+')
                state3(out_file, frame, phon, j, sent, cue);
            else
            .}
State 1, entered by a consonant (or vowel) at beginning of syllable */

void state1(FILE *out, int *fm, char [MAXLEN][4], int line, char name[1][10], char *cue)
{
    int i = line;
    int nwfm[3];
    /* DELETION */
    if (!(strcmp(l[i], "xxx")))
    {
        cue[i+1] = '*';
        /* Find which cue group the phonetic belongs to */
        /* Vowels */
        /* When the vowel group is found, you can immediately return the */
        /* playlist line using T hand shape and appropriate position */
        /* CHIN */
        else if (!(strcmp(l[i], "ao") || !(strcmp(l[i], "eh") ||
                        !(strcmp(l[i], "ux")))) ||
                        !(strcmp(l[i], "ao") || !(strcmp(l[i], "eh") ||
                        !(strcmp(l[i], "ux"))))
        {
            fprintf(out, "%s %0.5d %0.5d 31 7\n", name[0], fm[i], fm[i+1]);
            if (cue[i+1] == ',')
                cue[i+1] = '*';
        }
        /* THROAT */
        else if (!(strcmp(l[i], "ae") || !(strcmp(l[i], "ih") ||
                        !(strcmp(l[i], "ux")))) ||
                        !(strcmp(l[i], "ae") || !(strcmp(l[i], "ih") ||
                        !(strcmp(l[i], "ux"))))
        {
            fprintf(out, "%s %0.5d %0.5d 23 7\n", name[0], fm[i], fm[i+1]);
            if (cue[i+1] == ',')
                cue[i+1] = '*';
        }
        /* MOUTH */
        else if (!(strcmp(l[i], "iy") || !(strcmp(l[i], "er") ||
                        !(strcmp(l[i], "axr")))) ||
                        !(strcmp(l[i], "iy") || !(strcmp(l[i], "er") ||
                        !(strcmp(l[i], "axr"))))
        {
            fprintf(out, "%s %0.5d %0.5d 15 7\n", name[0], fm[i], fm[i+1]);
            if (cue[i+1] == ',')
                cue[i+1] = '*';
        }
        /* SIDE */
    }
    fclose(out_file);
}
else if (!(strcmp(l[i], "aa"))) || !(strcmp(l[i], "ow")))
    fprintf(out, "%s %0.5d %0.5d 7 7\n", name[0], fm[i], fm[i+1]);
    if (cue[i+1] == 's', 'r')
        cue[i+1] = 's';
*/ SIED */
else if (!(strcmp(l[i], "ah"))) || !(strcmp(l[i], "ax")))
    timeout(fm, nwfm, line, 1);
    fprintf(out, "%s %0.5d %0.5d 7 7\n", name[0], nwfm[0], nwfm[1]);
    fprintf(out, "%s %0.5d %0.5d 39 7\n", name[0], nwfm[1], nwfm[2]);
    if (cue[i+1] == 's', 'r')
        cue[i+1] = 's';
*/ DPT1 */
else if (!(strcmp(l[i], "ay"))) || !(strcmp(l[i], "aw")))
    timeout(fm, nwfm, line, 1);
    fprintf(out, "%s %0.5d %0.5d 7 7\n", name[0], nwfm[0], nwfm[1]);
    fprintf(out, "%s %0.5d %0.5d 23 7\n", name[0], nwfm[1], nwfm[2]);
    if (cue[i+1] == 's', 'r')
        cue[i+1] = 's';
*/ DPT2 */
else if (!(strcmp(l[i], "en")))
    timeout(fm, nwfm, line, 1);
    fprintf(out, "%s %0.5d %0.5d 31 7\n", name[0], nwfm[0], nwfm[1]);
    fprintf(out, "%s %0.5d %0.5d 5 5\n", name[0], nwfm[1], nwfm[2]);
    if (cue[i+1] == 's', 'r')
        cue[i+1] = 's';
*/ special case 'en' */
else if (!(strcmp(l[i], "en")))
    timeout(fm, nwfm, line, 1);
    fprintf(out, "%s %0.5d %0.5d 31 7\n", name[0], nwfm[0], nwfm[1]);
    fprintf(out, "%s %0.5d %0.5d 23 7\n", name[0], nwfm[1], nwfm[2]);
    if (cue[i+1] == 's', 'r')
        cue[i+1] = 's';
*/ Consonants */
/* When the consonant group is found you must then find out what */
/* vowels follows it (and possibly plosive) before the playlist */
/* line can be produced */
/* D GRP */
else if (!(strcmp(l[i], "d")) || !(strcmp(l[i], "dcl")) || !(strcmp(l[i], "dx")))
    !((strncpy(l[i], "dc1")) || !((strncpy(l[i], "dx")))
    !((strncpy(l[i], "pc1")) || !((strncpy(l[i], "zh")))
    !((strncpy(l[i], "dd")))
    state4(out, fm, 1, line+1, 0, name, cue);
/* G GRP */
else if (!(strcmp(l[i], "g")) || !(strcmp(l[i], "gc1")) ||
    !((strncpy(l[i], "gch"))) || !((strncpy(l[i], "th")))
    state4(out, fm, 1, line+1, 1, name, cue);
/* H GRP */
else if (!(strcmp(l[i], "h")) || !(strcmp(l[i], "s")) || !(strcmp(l[i], "r")))
    state4(out, fm, 1, line+1, 2, name, cue);
/* K GRP */
else if (!(strcmp(l[i], "k")) || !(strcmp(l[i], "kc1")) ||
    !((strncpy(l[i], "v"))) || !((strncpy(l[i], "dh"))) || !((strncpy(l[i], "z")))
    state4(out, fm, 1, line+1, 3, name, cue);
/* L GRP */

else if (!(strcmp(l[i], "l")) || !(strcmp(l[i], "sh"))) || !(strcmp(l[i], "w")))
    state4(out, fm, 1, line+1, 4, name, cue);
/* N GRP */
else if (!(strcmp(l[i], "n")) || !(strcmp(l[i], "b")) || !(strcmp(l[i], "bcl")))
    state4(out, fm, 1, line+1, 5, name, cue);
/* NG GRP */
else if (!(strcmp(l[i], "ng")) || !(strcmp(l[i], "y")) || !(strcmp(l[i], "ch")))
    state4(out, fm, 1, line+1, 6, name, cue);
/* T GRP */
else if (!(strcmp(l[i], "t")) || !(strcmp(l[i], "tcl")) || !(strcmp(l[i], "m")) || !(strcmp(l[i], "f")) || !(strcmp(l[i], "dt")))
    state4(out, fm, 1, line+1, 7, name, cue);

/* State2, entered by a consonant before another consonant at beginning of syllable */
void state2(FILE *out, int *fm, char l[MAXLEN][4], int line, char name[1][10], char *cue)
{
    int i = line;
    int nwfm[3];
    /* DELETION */
    if (!(strcmp(l[i], "xxx")))
        return;
    /* Find which consonant group the phonetics belong to */
    /* When found, can produce playlist line at the */
    /* side position */
    /* D GRP */
    if (!(strcmp(l[i], "d")) || !(strcmp(l[i], "dcl")) || !(strcmp(l[i], "dx")) || !(strcmp(l[i], "p")) || !(strcmp(l[i], "pcl")) || !(strcmp(l[i], "zh")) || !(strcmp(l[i], "dd")))
        fprintf(out, "%s %0.5d %0.5d 0 \n", name[0], fm[i], fm[i+1]);
    /* G GRP */
    else if (!(strcmp(l[i], "g")) || !(strcmp(l[i], "gc1")) || !(strcmp(l[i], "jh")) || !(strcmp(l[i], "th")))
        fprintf(out, "%s %0.5d %0.5d 1 1\n", name[0], fm[i], fm[i+1]);
    /* H GRP */
    else if (!(strcmp(l[i], "h")) || !(strcmp(l[i], "s")) || !(strcmp(l[i], "r")))
        fprintf(out, "%s %0.5d %0.5d 2 2\n", name[0], fm[i], fm[i+1]);
    /* K GRP */
    else if (!(strcmp(l[i], "k")) || !(strcmp(l[i], "kc1")) || !(strcmp(l[i], "v")) || !(strcmp(l[i], "dh")) || !(strcmp(l[i], "z")))
        fprintf(out, "%s %0.5d %0.5d 3 3\n", name[0], fm[i], fm[i+1]);
    /* L GRP */
    else if (!(strcmp(l[i], "l")) || !(strcmp(l[i], "sh")) || !(strcmp(l[i], "w")))
        fprintf(out, "%s %0.5d %0.5d 4 4\n", name[0], fm[i], fm[i+1]);
    /* N GRP */
    else if (!(strcmp(l[i], "n")) || !(strcmp(l[i], "b")) || !(strcmp(l[i], "bc1")))
        fprintf(out, "%s %0.5d %0.5d 5 5\n", name[0], fm[i], fm[i+1]);
    /* NG GRP */
    else if (!(strcmp(l[i], "ng")) || !(strcmp(l[i], "y")) || !(strcmp(l[i], "ch")))
        fprintf(out, "%s %0.5d %0.5d 6 6\n", name[0], fm[i], fm[i+1]);
    /* T GRP */
    else if (!(strcmp(l[i], "t")) || !(strcmp(l[i], "tc1")) || !(strcmp(l[i], "m")) || !(strcmp(l[i], "f")) || !(strcmp(l[i], "dt")))
        fprintf(out, "%s %0.5d %0.5d 7 7\n", name[0], fm[i], fm[i+1]);
/* Check for vowel groups in case a vowel was subed for a cons. */
/* CHIN */
else if (!(strcmp(l[i], "ao")) || !(strcmp(l[i], "eh")) ||
        !(strcmp(l[i], "uw")) || !(strcmp(l[i], "ux")))
    fprintf(out, "%s %0.5d %0.5d 31 7\n", name[0], fm[i], fm[i+1]);
/* THROAT */
else if (!(strcmp(l[i], "ae")) || !(strcmp(l[i], "ih")) ||
        !(strcmp(l[i], "ix")) || !(strcmp(l[i], "uh")))
    fprintf(out, "%s %0.5d %0.5d 23 7\n", name[0], fm[i], fm[i+1]);
/* MOUTH */
else if (!(strcmp(l[i], "iy")) || !(strcmp(l[i], "er")) ||
        !(strcmp(l[i], "axr")))
    fprintf(out, "%s %0.5d %0.5d 15 7\n", name[0], fm[i], fm[i+1]);
/* SIDE */
else if (!(strcmp(l[i], "aa")) || !(strcmp(l[i], "ow")))
    fprintf(out, "%s %0.5d %0.5d 7 7\n", name[0], fm[i], fm[i+1]);
/* SIDED */
else if (!(strcmp(l[i], "ah")) || !(strcmp(l[i], "ax")))
    {timeout(fm, nwfm, line, 1);
     fprintf(out, "%s %0.5d %0.5d 7 \n", name[0], nwfm[0], nwfm[1]);
     fprintf(out, "%s %0.5d %0.5d 23 \n", name[0], nwfm[1], nwfm[2]);
    } /* DPT1 */
else if (!(strcmp(l[i], "ay")) || !(strcmp(l[i], "aw")))
    {timeout(fm, nwfm, line, 1);
     fprintf(out, "%s %0.5d %0.5d 39 \n", name[0], nwfm[0], nwfm[1]);
     fprintf(out, "%s %0.5d %0.5d 23 \n", name[0], nwfm[1], nwfm[2]);
    } /* DPT2 */
else if (!(strcmp(l[i], "ey")) || !(strcmp(l[i], "oy")))
    {timeout(fm, nwfm, line, 1);
     fprintf(out, "%s %0.5d %0.5d 31 \n", name[0], nwfm[0], nwfm[1]);
     fprintf(out, "%s %0.5d %0.5d 23 \n", name[0], nwfm[1], nwfm[2]);
    } /* special case 'en' */
else if (!(strcmp(l[i], "en")))
    {timeout(fm, nwfm, line, 1);
     fprintf(out, "%s %0.5d %0.5d 31 \n", name[0], nwfm[0], nwfm[1]);
     fprintf(out, "%s %0.5d %0.5d 5\n", name[0], nwfm[1], nwfm[2]);
    }

/* State2A, entered by + a release of a consonant */
/* after a . a consonant before another at beginning of syllable */
void state2A(FILE *out, int *fm, char l[MAXLEN][4], int line, char name[1][10], char *cue)
{
    int i = line + 1;
    int nwfn[3];
    /* DELETION */
    if (!strcmp(l[i], "xxx"))
        return;
    /* Find which consonant group the phonetics belong to */
    /* When found, can produce playlist line at the */
    /* side position */
    /* D GRP */
    if (!(strcmp(l[i], "d")) || !(strcmp(l[i], "dc1")) || !(strcmp(l[i], "dx")) ||
        !(strcmp(l[i], "p")) || !(strcmp(l[i], "pc1")) || !(strcmp(l[i], "zh")) ||
        !(strcmp(l[i], "dd")))
        fprintf(out, "%s %0.5d %0.5d 0 0\n", name[0], fm[line], fm[i+1]);
    /* G GRP */
else if (!(strcmp(l[i],"g")) \|| (strcmp(l[i],"gc1")) \||
    !(strcmp(l[i],"gh")) \|| (strcmp(l[i],"th")))
    fprintf(out, "%s %0.5d %0.5d 0 1\n", name[0], fm[line], fm[i+1]);
    /* H GRP */
else if (!(strcmp(l[i],"h")) \|| (strcmp(l[i],"s")) \|| (strcmp(l[i],"r"))
    fprintf(out, "%s %0.5d %0.5d 0 2\n", name[0], fm[line], fm[i+1]);
    /* K GRP */
else if (!(strcmp(l[i],"k")) \|| (strcmp(l[i],"kcl")) \||
    !(strcmp(l[i],"v")) \|| (strcmp(l[i],"zh"))
    fprintf(out, "%s %0.5d %0.5d 0 3\n", name[0], fm[line], fm[i+1]);
    /* L GRP */
else if (!(strcmp(l[i],"n")) \|| (strcmp(l[i],"b")) \|| (strcmp(l[i],"bc1"))
    fprintf(out, "%s %0.5d %0.5d 0 4\n", name[0], fm[line], fm[i+1]);
    /* N GRP */
else if (!(strcmp(l[i],"ng")) \|| (strcmp(l[i],"y")) \|| (strcmp(l[i],"ch"))
    fprintf(out, "%s %0.5d %0.5d 0 5\n", name[0], fm[line], fm[i+1]);
    /* T GRP */
else if (!(strcmp(l[i],"t")) \|| (strcmp(l[i],"tcl")) \|| (strcmp(l[i],"m")) \||
    !(strcmp(l[i],"f")) \|| (strcmp(l[i],"dt"))
    fprintf(out, "%s %0.5d %0.5d 0 6\n", name[0], fm[line], fm[i+1]);
    /* DPT1 */
else if (!(strcmp(l[i],"ay")) \|| (strcmp(l[i],"aw"))
    fprintf(out, "%s %0.5d %0.5d 0 7\n", name[0], fm[line], fm[i+1]);
    /* DPT2 */
else if (!(strcmp(l[i],"ey")) \|| (strcmp(l[i],"oy"))
    fprintf(out, "%s %0.5d %0.5d 0 8\n", name[0], fm[line], fm[i+1]);
    /* special case 'em' */
else if (!(strcmp(l[i],"en")))
{
    timeout(fm, nwfm, i, 2);
    fprintf(out, "%s %0.5d %0.5d 31 7\n", name[0], nwfm[0], nwfm[1]);
    fprintf(out, "%s %0.5d %0.5d 5 5\n", name[0], nwfm[1], nwfm[2]);
}

/* State3, entered by + a release of a plosive after a ' beginning of a syllable */
void state3(FILE *out, int *fm, char I[MAXLEN][4], int line, char name[1][10], char *cue) 360
{
    int i = line + 1;
    int nwfm[3];
    /* DELETION */
    if (!(strcmp(l[i],"xxx")))
    {
        cue[i+1] = ' *';
        / * D GRP */
    else if (!(strcmp(l[i],"d")) || !(strcmp(l[i],"dcl")) || !(strcmp(l[i],"dx")) ||
                !(strcmp(l[i],"p")) || !(strcmp(l[i],"pcl")) || !(strcmp(l[i],"zh")) ||
                !(strcmp(l[i],"dd")))
        state5(out, fm, 1, line+2, 0, name, cue);
        / * G GRP */
    else if (!(strcmp(l[i],"g")) || !(strcmp(l[i],"gc1")) ||
                !(strcmp(l[i],"jh")) || !(strcmp(l[i],"th")))
        state5(out, fm, 1, line+2, 1, name, cue);
        / * H GRP */
    else if (!(strcmp(l[i],"h")) || !(strcmp(l[i],"s")) || !(strcmp(l[i],"r"))
        state5(out, fm, 1, line+2, 2, name, cue);
        / * K GRP */
    else if (!(strcmp(l[i],"k")) || !(strcmp(l[i],"kcl")) ||
                !(strcmp(l[i],"v")) || !(strcmp(l[i],"zh")) ||!(strcmp(l[i],"z")))
        state5(out, fm, 1, line+2, 3, name, cue);
        / * L GRP */
    else if (!(strcmp(l[i],"l")) || !(strcmp(l[i],"sh")) || !(strcmp(l[i],"w")))
        state5(out, fm, 1, line+2, 4, name, cue);
        / * N GRP */
    else if (!(strcmp(l[i],"n")) || !(strcmp(l[i],"b")) || !(strcmp(l[i],"bcl")))
        state5(out, fm, 1, line+2, 5, name, cue);
        / * NG GRP */
    else if (!(strcmp(l[i],"ng")) || !(strcmp(l[i],"y")) || !(strcmp(l[i],"ch")))
        state5(out, fm, 1, line+2, 6, name, cue);
        / * T GRP */
    else if (!(strcmp(l[i],"t")) || !(strcmp(l[i],"tc1")) || !(strcmp(l[i],"m")) ||
                !(strcmp(l[i],"f")) || !(strcmp(l[i],"dt")))
        state5(out, fm, 1, line+2, 7, name, cue);
        /* Need to check for vowel group in case vowel was subed for cons */
        / * CHIN */
    else if (!(strcmp(l[i],"ao")) || !(strcmp(l[i],"eh")) ||
                !(strcmp(l[i],"uw")) || !(strcmp(l[i],"ux")))
        {fprintf(out, "%s %0.5d %0.5d 31 7\n", name[0], fm[line], fm[i+1]);
            cue[i+1] = ' *';}
        / * THROAT */
    else if (!(strcmp(l[i],"ae")) || !(strcmp(l[i],"ih")) ||
                !(strcmp(l[i],"ix")) || !(strcmp(l[i],"uh")))
        {fprintf(out, "%s %0.5d %0.5d 23 7\n", name[0], fm[line], fm[i+1]);
            cue[i+1] = ' *';}
        / * MOUTH */
}
else if (!(strcmp(l[i],"iy")) || !(strcmp(l[i],"er")) || !(strcmp(l[i],"axr")))
  {fprintf(out, "%s %0.5d %0.5d 15 \n", name[0], fm[line], fm[i+1]);
    cue[i+1] = '*' ;}
 /* SIDE */
else if (!(strcmp(l[i],"aa") || !(strcmp(l[i],"ow")))
  {fprintf(out, "%s %0.5d %0.5d 7 7
", name[0], fm[line], fm[i+1]);
    cue[i+1] = '*';}
/* SIDE */
else if (!(strcmp(l[i],"ah") || !(strcmp(l[i],"ax")))
  {timeout(fm, nwfm, i, 2);
    fprintf(out, "%s %0.5d %0.5d 7 \n", name[0], nwfm[0], nwfm[1]);
    fprintf(out, "%s %0.5d %0.5d 39 \n", name[0], nwfm[1], nwfm[2]);
    cue[i+1] = '*';}
/* DPT1 */
else if (!(strcmp(l[i],"ay") || !(strcmp(l[i],"aw")))
  {timeout(fm, nwfm, i, 2);
    fprintf(out, "%s %0.5d %0.5d 31 7
", name[0], nwfm[0], nwfm[1]);
    fprintf(out, "%s %0.5d .05d 23 7
", name[0], nwfm[1], nwfm[2]);
    cue[i+1] = '*';}
/* DPT2 */
else if (!(strcmp(l[i],"ey") || !(strcmp(l[i],"oy")))
  {timeout(fm, nwfm, i, 2);
    fprintf(out, "%s %0.5d %0.5d 31 7
", name[0], nwfm[0], nwfm[1]);
    fprintf(out, "%s %0.5d %0.5d 5 5
", name[0], nwfm[1], nwfm[2]);
    cue[i+1] = '*';}
/* special case 'en' */
else if (!(strcmp(l[i],"en"))
  {timeout(fm, nwfm, i, 2);
    fprintf(out, "%s %0.5d %0.5d 31 7
", name[0], nwfm[0], nwfm[1]);
    fprintf(out, "%s %0.5d .05d 23 7
", name[0], nwfm[1], nwfm[2]);
    cue[i+1] = '*';}
    /* State4, entered by , a vowel after a consonant */
void state4(FILE *out, int *fm, char [MAXLEN][4], int line, int shp, char name[1][10], char *cue)
  {
    i = line;
    int pos, pos2;
    int nwfm[3];
    /* DELETION */
    if (!(strcmp(l[i],"xxx") || (cue[i] == ', '))
      fprintf(out, "%s %0.5d %0.5d %d \n", name[0], fm[i-1], fm[i], shp);
    /* Find which vowel group the phonetic belongs to */
    /* When vowel group is found, use it in conjunction with the */
    /* hand shape passed in to produce playlist line */
    /* CHIN */
    else if (!(strcmp(l[i],"oa") || !(strcmp(l[i],"eh")) ||
            !(strcmp(l[i],"uw") || !(strcmp(l[i],"ux"))))
    {switch(shp)
      {case 0: pos = 24; break;
        case 1: pos = 25; break;
        case 2: pos = 26; break;
        case 3: pos = 27; break;
        case 4: pos = 28; break;
        case 5: pos = 29; break;
}
case 6: pos = 30; break;
case 7: pos = 31; break;
default: break;
}
fprintf(out, "%s %0.5d %0.5d %d %d
", name[0], fm[i-1], fm[i+1], pos, shp);
}
/* THROAT */
else if (!(strcmp(l[i],"ae")) || !(strcmp(l[i],"ih")) ||
!(strcmp(l[i],"ix")) || !(strcmp(l[i],"uh")))
{switch(shp)
  {case 0: pos = 16; break;
case 1: pos = 17; break;
case 2: pos = 18; break;
case 3: pos = 19; break;
case 4: pos = 20; break;
case 5: pos = 21; break;
case 6: pos = 22; break;
case 7: pos = 23; break;
default: break; }
fprintf(out, "%s %0.5d %0.5d %d %d
", name[0], fm[i-1], fm[i+1], pos, shp);
}
/* MOUTH */
else if (!(strcmp(l[i],"iy")) || !(strcmp(l[i],"er")) || !(strcmp(l[i],"axr")))
{switch(shp)
  {case 0: pos = 8; break;
case 1: pos = 9; break;
case 2: pos = 10; break;
case 3: pos = 11; break;
case 4: pos = 12; break;
case 5: pos = 13; break;
case 6: pos = 14; break;
case 7: pos = 15; break;
default: break; }
fprintf(out, "%s %0.5d %0.5d %d %d
", name[0], fm[i-1], fm[i+1], pos, shp);
}
/* SIDE */
else if (!(strcmp(l[i],"aa")) || !(strcmp(l[i],"ow")))
{switch(shp)
  {case 0: pos = 0; break;
case 1: pos = 1; break;
case 2: pos = 2; break;
case 3: pos = 3; break;
case 4: pos = 4; break;
case 5: pos = 5; break;
case 6: pos = 6; break;
case 7: pos = 7; break;
default: break; }
fprintf(out, "%s %0.5d %0.5d %d %d
", name[0], fm[i-1], fm[i+1], pos, shp);
}
/* SIDED */
else if (!(strcmp(l[i],"ah")) || !(strcmp(l[i],"ax")))
{switch(shp)
  {case 0: pos = 0; pos2 = 32; break;
case 1: pos = 1; pos2 = 33; break;
case 2: pos = 2; pos2 = 34; break;
  }
case 3: pos = 3; pos2 = 35; break;
case 4: pos = 4; pos2 = 36; break;
case 5: pos = 5; pos2 = 37; break;
case 6: pos = 6; pos2 = 38; break;
case 7: pos = 7; pos2 = 39; break;
default: break; }

timeout(fm, nwfm, line, 2);
fprintf(out, "%s %0.5d %0.5d %d\n", name[0], nwfm[0], nwfm[1], pos, shp);
fprintf(out, "%s %0.5d %0.5d %d\n", name[0], nwfm[1], nwfm[2], pos2, shp);
}

/* DPT1 */
else if (!(strcmp(l[i], "ay")) || !(strcmp(l[i], "aw")))
{
switch(shp)
{
case 0: pos = 0; break;
case 1: pos = 1; break;
case 2: pos = 2; break;
case 3: pos = 3; break;
case 4: pos = 4; break;
case 5: pos = 5; break;
case 6: pos = 6; break;
case 7: pos = 7; break;
default: break; }

timeout(fm, nwfm, line, 2);
fprintf(out, "%s %0.5d %0.5d %d\n", name[0], nwfm[0], nwfm[1], pos, shp);
fprintf(out, "%s %0.5d %0.5d %d\n", name[0], nwfm[1], nwfm[2], pos2, shp);
}

/* DPT2 */
else if (!(strcmp(l[i], "ey")) || !(strcmp(l[i], "oy")))
{
switch(shp)
{
case 0: pos = 24; break;
case 1: pos = 25; break;
case 2: pos = 26; break;
case 3: pos = 27; break;
case 4: pos = 28; break;
case 5: pos = 29; break;
case 6: pos = 30; break;
case 7: pos = 31; break;
default: break; }

timeout(fm, nwfm, line, 2);
fprintf(out, "%s %0.5d %0.5d %d\n", name[0], nwfm[0], nwfm[1], pos, shp);
fprintf(out, "%s %0.5d %0.5d 23 7\n", name[0], nwfm[1], nwfm[2]);
}

/* special case 'en' */
else if (!(strcmp(l[i], "en")))
{
switch(shp)
{
case 0: pos = 24; break;
case 1: pos = 25; break;
case 2: pos = 26; break;
case 3: pos = 27; break;
case 4: pos = 28; break;
case 5: pos = 29; break;
case 6: pos = 30; break;
case 7: pos = 31; break;
default: break; }

timeout(fm, nwfm, line, 2);
fprintf(out, "%s %0.5d %0.5d %d\n", name[0], nwfm[0], nwfm[1], pos, shp);
fprintf(out, "%s %0.5d %0.5d 23 7\n", name[0], nwfm[1], nwfm[2]);
}

/* special case 'en' */
timeout(fm, nwfm, line, 2);
fprintf(out, "%s %0.5d %0.5d %d \n", name[0], nwfm[0], nwfm[1], pos, shp);
fprintf(out, "%s %0.5d %0.5d %0.5d %d %d\n", name[0], nwfm[1], nwfm[2]);
}
/* Vowel -> Consonant */
else
    {fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], nwfm[0], nwfm[1], pos, shp);
    cue[i] = '*';}
}

/* State5, entered by a vowel after a consonant */
void state5(FILE *out, int *fm, char l[MAXLEN][4], int line, int shp, char name[1][10], char &cue)
{
    int i = line;
    int pos, pos2;
    int nwfm[3];
/* DELETION */
    if (!(strcmp(l[i], "xxx") || (cue[i] != '"')))
        {fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], fm[i-2], fm[i], shp, shp);
        return;}
/* Find which vowel group the phonetic belongs to */
/* When vowel group is found, use it in conjunction with the */
/* hand shape passed in to produce playlist line */
/* CHIN */
else if (!(strcmp(l[i], "ao") || !(strcmp(l[i], "eh") ||
                      !(strcmp(l[i], "uw") || !(strcmp(l[i], "ux"))))
    {switch(shp)
      {case 0: pos = 24; break;
       case 1: pos = 25; break;
       case 2: pos = 26; break;
       case 3: pos = 27; break;
       case 4: pos = 28; break;
       case 5: pos = 29; break;
       case 6: pos = 30; break;
       case 7: pos = 31; break;
       default: break; }
    fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], fm[i-2], fm[i], pos, shp);
    }
/* THROAT */
else if (!((strcmp(l[i], "ae") || !(strcmp(l[i], "ih") ||
                      !((strcmp(l[i], "ix") || !(strcmp(l[i], "uh"))))))
    {switch(shp)
      {case 0: pos = 16; break;
       case 1: pos = 17; break;
       case 2: pos = 18; break;
       case 3: pos = 19; break;
       case 4: pos = 20; break;
       case 5: pos = 21; break;
       case 6: pos = 22; break;
       case 7: pos = 23; break;
       default: break; }
    fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], fm[i-2], fm[i], pos, shp);
    }
/* MOUTH */
else if (!((strcmp(l[i], "iy") || !(strcmp(l[i], "er") ||
                      !((strcmp(l[i], "axr") || !(strcmp(l[i], "axr"))))
    {switch(shp)
      {case 0: pos = 16; break;
       case 1: pos = 17; break;
       case 2: pos = 18; break;
       case 3: pos = 19; break;
       case 4: pos = 20; break;
       case 5: pos = 21; break;
       case 6: pos = 22; break;
       case 7: pos = 23; break;
       default: break; }
    fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], fm[i-2], fm[i+1], pos, shp);
    }
/* MOUTH */
else if (!((strcmp(l[i], "iy") || !(strcmp(l[i], "er") ||
                      !((strcmp(l[i], "axr") || !(strcmp(l[i], "axr"))))
    {switch(shp)
      {case 0: pos = 16; break;
       case 1: pos = 17; break;
       case 2: pos = 18; break;
       case 3: pos = 19; break;
       case 4: pos = 20; break;
       case 5: pos = 21; break;
       case 6: pos = 22; break;
       case 7: pos = 23; break;
       default: break; }
    fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], fm[i-2], fm[i+1], pos, shp);
    }
/* MOUTH */
else if (!((strcmp(l[i], "iy") || !(strcmp(l[i], "er") ||
                      !((strcmp(l[i], "axr") || !(strcmp(l[i], "axr"))))
    {switch(shp)
      {case 0: pos = 16; break;
       case 1: pos = 17; break;
       case 2: pos = 18; break;
       case 3: pos = 19; break;
       case 4: pos = 20; break;
       case 5: pos = 21; break;
       case 6: pos = 22; break;
       case 7: pos = 23; break;
       default: break; }
    fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], fm[i-2], fm[i+1], pos, shp);
    }
/* MOUTH */
else if (!((strcmp(l[i], "iy") || !(strcmp(l[i], "er") ||
                      !((strcmp(l[i], "axr") || !(strcmp(l[i], "axr"))))
    {switch(shp)
      {case 0: pos = 16; break;
       case 1: pos = 17; break;
       case 2: pos = 18; break;
       case 3: pos = 19; break;
       case 4: pos = 20; break;
       case 5: pos = 21; break;
       case 6: pos = 22; break;
       case 7: pos = 23; break;
       default: break; }
    fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], fm[i-2], fm[i+1], pos, shp);
    }
{switch(shp)
    {case 0: pos = 8; break;
    case 1: pos = 9; break;
    case 2: pos = 10; break;
    case 3: pos = 11; break;
    case 4: pos = 12; break;
    case 5: pos = 13; break;
    case 6: pos = 14; break;
    case 7: pos = 15; break;
    default: break; }
    fprintf(out, "s %0.5d %0.5d %d \n", name[0], fm[i-2], fm[i+1], pos, shp);
}
/* SIDE */
else if (!(strcmp(l[i],"aa")) || !(strcmp(l[i],"ow")))
{switch(shp)
    {case 0: pos = 0; break;
    case 1: pos = 1; break;
    case 2: pos = 2; break;
    case 3: pos = 3; break;
    case 4: pos = 4; break;
    case 5: pos = 5; break;
    case 6: pos = 6; break;
    case 7: pos = 7; break;
    default: break; }
    fprintf(out, "%s %0.5d %0.5d %d
", name[0], fm[i-2], fm[i+1], pos, shp);
}
/* SIDED */
else if (!(strcmp(l[i],"ah")) || !(strcmp(l[i],"ax")))
{switch(shp)
    {case 0: pos = 0; pos2 = 32; break;
    case 1: pos = 1; pos2 = 33; break;
    case 2: pos = 2; pos2 = 34; break;
    case 3: pos = 3; pos2 = 35; break;
    case 4: pos = 4; pos2 = 36; break;
    case 5: pos = 5; pos2 = 37; break;
    case 6: pos = 6; pos2 = 38; break;
    case 7: pos = 7; pos2 = 39; break;
    default: break; }
timeout(fm, nwfm, line, 3);
    fprintf(out, "%s %0.5d %0.5d %d \n", name[0], nwfm[0], nwfm[1], pos, shp);
    fprintf(out, "%s %0.5d %0.5d %d %d
", name[0], nwfm[1], nwfm[2], pos2, shp);
}
/* DPT1 */
else if (!(strcmp(l[i],"ay")) || !(strcmp(l[i],"aw")))
{switch(shp)
    {case 0: pos = 0; break;
    case 1: pos = 1; break;
    case 2: pos = 2; break;
    case 3: pos = 3; break;
    case 4: pos = 4; break;
    case 5: pos = 5; break;
    case 6: pos = 6; break;
    case 7: pos = 7; break;
    default: break; }
timeout(fm, nwfm, line, 3);
fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], nwfm[0], nwfm[1], pos, shp);
fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], nwfm[1], nwfm[2]);
}
/* DPT2 */
} else if (!strcmp(l[i], "ey") || !strcmp(l[i], "oy"))
{switch(shp)
  {case 0: pos = 24; break;
   case 1: pos = 25; break;
   case 2: pos = 26; break;
   case 3: pos = 27; break;
   case 4: pos = 28; break;
   case 5: pos = 29; break;
   case 6: pos = 30; break;
   case 7: pos = 31; break;
   default: break; }
}
/* special case 'en' */
} else if (!strcmp(l[i], "en")
{switch(shp)
  {case 0: pos = 24; break;
   case 1: pos = 25; break;
   case 2: pos = 26; break;
   case 3: pos = 27; break;
   case 4: pos = 28; break;
   case 5: pos = 29; break;
   case 6: pos = 30; break;
   case 7: pos = 31; break;
   default: break; }
}
/* Vowel -> Consonant */
else
{fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], fm[i-2], fm[i], shp, shp);
cue[i] = ' *';}
}

/*****************************************************************
/* Function timeout divides the frame duration into two equal parts which is needed for diphthongs and side--down cues */
/* Opt 1 is used when only 1 line needs to be divided up */
/* i.e. for a vowel by itself */
/* Opt 2 is used when 2 lines need to be divided up */
/* i.e. for a consonant followed by a vowel */
/* Opt 3 is used when 3 lines need to be divided up */
/* i.e. for a consonant followed by a plosive followed by a vowel */
*******************************************************************/
void timeout(int *tm, int *frm, int line, int opt)
{ int temp;
if (opt == 1)
    {frm[0] = tm[line];
     temp = (tm[line+1] - tm[line]) / 2;
     frm[1] = tm[line] + temp;
     frm[2] = tm[line+1];
    }

if (opt == 2)
    {frm[0] = tm[line-1];
     temp = (tm[line+1] - tm[line-1]) / 2;
     frm[1] = tm[line-1] + temp;
     frm[2] = tm[line+1];
    }

if (opt == 3)
    {frm[0] = tm[line-2];
     temp = (tm[line+1] - tm[line-2]) / 2;
     frm[2] = tm[line+1];
    }

The only difference between State and the versions that introduce delay is the function frameout. All other code is identical. The frameout function from statedl.c is listed here as an example of fixed frame delay. It is followed by the frameout function from randst.c as an example of random delay.
int frameout(double *tm, int *frm, int start, int stop)
{
    int i = 0;

    while (tm[i] != '\0')
    {
        tm[i] = tm[i] - 530; /* correct for pcm error */
        tm[i] = tm[i] * f;   /* correct for dyaxis error */
        frm[i] = ((int) (0.5 + tm[i]/1000*30)) + start; /* change to frames */
        frm[i] = frm[i] + 1; /* add one frame delay (or 3 or 5) */
        i++;
    }
    if ((frm[0] < start) || (frm[i-1] > stop))
        printf("ERROR: OVERWRITING BLUE\n");
        return(0);
    return(1);
}

The version of State that marks some cues with red box outlines to signify possibly erroneous cues is listed in its entirety below due to subtle differences between it and state.c.
**statemark.c**

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <math.h>
#define MAXLEN 80 /* maximum number of lines in input file */
#define f .96432 /* correction factor for sampling */

void state1(FILE *, int *, char phon[MAXLEN][4], int, char sent[1][10], char *, int *);
void state2(FILE *, int *, char phon[MAXLEN][4], int, char sent[1][10], char *, int *);
void state2A(FILE *, int *, char phon[MAXLEN][4], int, char sent[1][10], char *, int *);
void state3(FILE *, int *, char phon[MAXLEN][4], int, char sent[1][10], char *, int *);
void state4(FILE *, int *, char phon[MAXLEN][4], int, int, char sent[1][10], char *, int *);
void state5(FILE *, int *, char phon[MAXLEN][4], int, int, char sent[1][10], char *, int *);
void timeout(int *, int *, int, int);
int frameout(double *, int *, int, int);
void errmark(char err[MAXLEN][5], int *);

main(int argc, char *argv[])
{
    FILE *sent_file; /* pointer to file to read in */
    FILE *frame_file; /* pointer to file that has frame info */
    FILE *out_file; /* pointer to file to put output */
    char filename[30]; /* name of file to read in */
    char phon[MAXLEN][4]; /* 2D array to hold phonemes lineXphon. length */
    char sent[1][10]; /* array for sentence id */
    char fmt[MAXLEN]; /* array to hold time values */
    char cue[MAXLEN]; /* array to hold cue markings */
    int frame[MAXLEN]; /* array to hold absolute frames numbers */
    int i = 0, j = 0, k; /* array index */
    char err[MAXLEN][5]; /* array to hold error markings */
    char line[20]; /* array to hold each line of file */
    int qm[MAXLEN]; /* indicate that the cue may have an error */

    if (argc == 1)
    {
        printf("Input name of sentence file> ");
        scanf("%s", filename);
        sent_file = fopen(filename, "r");
        out_file = stdout;
    }
    else if (argc == 2)
    {
        sent_file = fopen(argv[1], "r");
        out_file = stdout;
    }
    else if (argc == 3)
    {  

```

{ sent_file = fopen(argv[1], "r");
  out_file = fopen(argv[2], "w"); }

/* Divide file components (times, cues, phonemes) into separate arrays */
while (fgets(line,40,sent_file) != NULL)
  {sscanf(line, "%lf/%c%3s %*s %s", &ftm[i], &cue[i], phon[i], err[i]);
    i++;}
fclose(sent_file);

/* Calls function to figure out which phones have errors */
ermark(err, qm, i-1);

/* Reads file that contains information on sentence */
/* and start and stop frames for the sentence (nonblue) */
frame_file = fopen("finfo.dat", "r");
if (fscanf(frame-file, "%s%d%d", sent[0], &startfm, &stopfm) != EOF)
  ;
fclose(frame_file);

/* Convert times to frame values. Exit if overwriting into blue. */
if ((frameout(ftm, frame, startfm, stopfm)) == 0)
  printf("IN SENTENCE %s\n", sent[0]);

fprintf(out_file, "\nSENTENCES
{\n");
fprintf(outfile, "%s %0.5d %Y0.5d\n\n\nPLAYLIST
{\n");

/* Implement the state diagram */
for (j=0; cue[j] != '\0'; j++)
  {
switch(cue[j])
    {case ' - ':
      break;
    case ' . ':
      if (cue[j+1] == ' + ')
        state3 (out_file, frame, phon, j, sent, cue, qm);
      else
        state1 (out_file, frame, phon, j, sent, cue, qm);
      break;
    case '*':
      if (cue[j+1] == ' + ')
        state2A (out_file, frame, phon, j, sent, cue, qm);
      else
        state2 (out_file, frame, phon, j, sent, cue, qm);
      break;
    case ',':
      break;
    case '+':
      break;
    default:
      break;
  }
default:
    break;
}

fprintf(out_file, ";n");
fclose(out_file);

/* State 1, entered by 'a consonant (or vowel) at beginning of syllable */
void state1(FILE *out, int *fm, char l[MAXLEN][4], int line, char name[1][10], char *cue, int *qm)
{
    int i = line;
    int nwf[3];
    /* DELETION */
    if (!strcmp(l[i], "xxx"))
        cue[i+1] = '*';
    /* Find which cue group the phonetic belongs to */
    /* Vowels */
    /* When the vowel group is found, you can immediately return the */
    /* playlist line using T hand shape and appropriate position */
    /* CHIN */
    else if (!strcmp(l[i], "ao") || !strcmp(l[i], "eh") ||
             !strcmp(l[i], "ux") || !strcmp(l[i], "ux"))
    {
        fprintf(out, "%s %0.5d %0.5d 31 %d7
", name[i], fm[i], fm[i+1], qm[i]);
        if (cue[i+1] == ',')
            cue[i+1] = '*';
    }
    /* THROAT */
    else if (!strcmp(l[i], "ae") || !strcmp(l[i], "ih") ||
             !strcmp(l[i], "ax") || !strcmp(l[i], "uh"))
    {
        fprintf(out, "%s %0.5d %0.5d 23 %d7
", name[i], fm[i], fm[i+1], qm[i]);
        if (cue[i+1] == ',')
            cue[i+1] = '*';
    }
    /* MOUTH */
    else if (!strcmp(l[i], "iy") || !strcmp(l[i], "er") || !strcmp(l[i], "axr"))
    {
        fprintf(out, "%s %0.5d %0.5d 15 %d7
", name[i], fm[i], fm[i+1], qm[i]);
        if (cue[i+1] == ',')
            cue[i+1] = '*';
    }
    /* SIDE */
    else if (!strcmp(l[i], "aa") || !strcmp(l[i], "ow"))
    {
        fprintf(out, "%s %0.5d %0.5d 7 %d7
", name[i], fm[i], fm[i+1], qm[i]);
        if (cue[i+1] == ',')
            cue[i+1] = '*';
    }
    /* SIVED */
    else if (!strcmp(l[i], "ah") || !strcmp(l[i], "ax"))
    {
        timeout(fm, nwf, line, 1);
        fprintf(out, "%s %0.5d %0.5d 7 %d7
", name[i], nwfm[0], nwfm[1], qm[i]);
        fprintf(out, "%s %0.5d %0.5d 23 %d7
", name[i], nwfm[1], nwfm[2], qm[i]);
        if (cue[i+1] == ',')
            cue[i+1] = '*';
    }
    /* DPT1 */
    else if (!strcmp(l[i], "ay") || !strcmp(l[i], "aw"))
    {
        timeout(fm, nwf, line, 1);
        fprintf(out, "%s %0.5d %0.5d 7 %d7
", name[i], nwfm[0], nwfm[1], qm[i]);
        fprintf(out, "%s %0.5d %0.5d 23 %d7
", name[i], nwfm[1], nwfm[2], qm[i]);
        if (cue[i+1] == ',')
            cue[i+1] = '*';
    }
}
else if (!strcmp(l[i],"ey") || !strcmp(l[i],"oy"))
    {timeout(fm, nwfm, line, 1);
        fprintf(out, "%s %0.5d %0.5d 31 %d7\n", name[0], nwfm[0], nwfm[1],qm[i]);
        fprintf(out, "%s %0.5d %0.5d 23 %d7\n", name[0], nwfm[1], nwfm[2],qm[i]);
        if (cue[i+1] == ', ')
            cue[i+1] = '*';
        /* special case 'en' */
    }

/* Consonants */
/* When the consonant group is found you must then find out what */
/* vowels follows it (and possibly plosive) before the playlist */
/* line can be produced */
/* D GRP */
else if (!strcmp(l[i],"d") || !strcmp(l[i],"dcl") || !strcmp(l[i],"dx") ||
         !strcmp(l[i],"p") || !strcmp(l[i],"pcl") || !strcmp(l[i],"zh") ||
         !strcmp(l[i],"dd"))
    state4(out, fm, 1, line+l, 0, name, cue, qm);
/* G GRP */
else if (!strcmp(l[i],"g") || !strcmp(l[i],"gcl") ||
         !strcmp(l[i],"jh") || !strcmp(l[i],"ch") ||
         !strcmp(l[i],"dd"))
    state4(out, fm, 1, line+l, 1, name, cue, qm);
/* H GRP */
else if (!strcmp(l[i],"h") || !strcmp(l[i],"s") || !strcmp(l[i],"r") ||
         !strcmp(l[i],"sh") || !strcmp(l[i],"w") ||
         !strcmp(l[i],"zh") || !strcmp(l[i],"z"))
    state4(out, fm, 1, line+l, 2, name, cue, qm);
/* K GRP */
else if (!strcmp(l[i],"k") || !strcmp(l[i],"kcl") ||
         !strcmp(l[i],"v") || !strcmp(l[i],"dh") || !strcmp(l[i],"z"))
    state4(out, fm, 1, line+l, 3, name, cue, qm);
/* L GRP */
else if (!strcmp(l[i],"l") || !strcmp(l[i],"sh") || !strcmp(l[i],"w"))
    state4(out, fm, 1, line+l, 4, name, cue, qm);
/* N GRP */
else if (!strcmp(l[i],"n") || !strcmp(l[i],"b") || !strcmp(l[i],"bc1"))
    state4(out, fm, 1, line+l, 5, name, cue, qm);
/* NG GRP */
else if (!strcmp(l[i],"ng") || !strcmp(l[i],"y") || !strcmp(l[i],"ch"))
    state4(out, fm, 1, line+l, 6, name, cue, qm);
/* T GRP */
else if (!strcmp(l[i],"t") || !strcmp(l[i],"tcl") || !strcmp(l[i],"m") ||
         !strcmp(l[i],"f") || !strcmp(l[i],"dt"))
    state4(out, fm, 1, line+l, 7, name, cue, qm);
}

/* State2, entered by a consonant before another consonant at beginning of syllable */
void state2(FILE *out, int *fm, char l[MAXLEN][4], int line, char name[1][10], char *cue, int *qm)
{ int i = line;
    int nwfm[3];
DELETION */
if (!(strcmp(l[i], "xxx")))
    return;
/
* Find which consonant group the phonetics belong to */
* When found, can produce playlist line at the */
* side position */
* D GRP */
if (!(strcmp(l[i], "d")) || !(strcmp(l[i], "dc") ) || !(strcmp(l[i], "dx")) ) ||
!(strcmp(l[i], "p") ) || !(strcmp(l[i], "pc") ) || !(strcmp(l[i], "zh") ) ||
!(strcmp(l[i], "dd") )
fprintf(out, "%s %0.5d %0.5d 0 %d\n", name[0], fm[i], fm[i+1], qm[i]);
/* G GRP */
else if ( !(strcmp(l[i], "g") ) || !(strcmp(l[i], "gc") ) ) ||
!(strcmp(l[i], "jh") ) || !(strcmp(l[i], "th") )
fprintf(out, "%s %0.5d %0.5d 1 %d\n", name[0], fm[i], fm[i+1], qm[i]);
/* H GRP */
else if ( !(strcmp(l[i], "h") ) || !(strcmp(l[i], "s") ) || !(strcmp(l[i], "r") )
fprintf(out, "%s %0.5d %0.5d 2 %d\n", name[0], fm[i], fm[i+1], qm[i]);
/* K GRP */
else if ( !(strcmp(l[i], "k") ) || !(strcmp(l[i], "kc") ) ||
!(strcmp(l[i], "zh") ) || !(strcmp(l[i], "z") )
fprintf(out, "%s %0.5d %0.5d 3 %d\n", name[0], fm[i], fm[i+1], qm[i]);
/* L GRP */
else if ( !(strcmp(l[i], "n") ) || !(strcmp(l[i], "b") ) || !(strcmp(l[i], "bc") )
fprintf(out, "%s %0.5d %0.5d 4 %d\n", name[0], fm[i], fm[i+1], qm[i]);
/* N GRP */
else if ( !(strcmp(l[i], "ng") ) || !(strcmp(l[i], "y") ) || !(strcmp(l[i], "ch") )
fprintf(out, "%s %0.5d %0.5d 5 %d\n", name[0], fm[i], fm[i+1], qm[i]);
/* NG GRP */
else if ( !(strcmp(l[i], "ng") ) || !(strcmp(l[i], "y") ) || !(strcmp(l[i], "ch") )
fprintf(out, "%s %0.5d %0.5d 6 %d\n", name[0], fm[i], fm[i+1], qm[i]);
/* T GRP */
else if ( !(strcmp(l[i], "t") ) || !(strcmp(l[i], "tc") ) || !(strcmp(l[i], "m") ) ||
!(strcmp(l[i], "f") ) || !(strcmp(l[i], "dt") )
fprintf(out, "%s %0.5d %0.5d 7 %d\n", name[0], fm[i], fm[i+1], qm[i]);
/* CHIN */
else if ( !(strcmp(l[i], "ao") ) || !(strcmp(l[i], "eh") ) ||
!(strcmp(l[i], "uw") ) || !(strcmp(l[i], "ux") )
fprintf(out, "%s %0.5d %0.5d 8 %d\n", name[0], fm[i], fm[i+1], qm[i]);
/* THROAT */
else if ( !(strcmp(l[i], "ae") ) || !(strcmp(l[i], "ih") ) ||
!(strcmp(l[i], "ix") ) || !(strcmp(l[i], "uh") )
fprintf(out, "%s %0.5d %0.5d 9 %d\n", name[0], fm[i], fm[i+1], qm[i]);
/* MOUTH */
else if ( !(strcmp(l[i], "iy") ) || !(strcmp(l[i], "ez") ) || !(strcmp(l[i], "axr") )
fprintf(out, "%s %0.5d %0.5d 10 %d\n", name[0], fm[i], fm[i+1], qm[i]);
/* SIDE */
else if ( !(strcmp(l[i], "aa") ) || !(strcmp(l[i], "ow") )
fprintf(out, "%s %0.5d %0.5d 11 %d\n", name[0], fm[i], fm[i+1], qm[i]);
/* SIDED */
else if ( !(strcmp(l[i], "ah") ) || !(strcmp(l[i], "ax") )
{timeout(fm, nwfm, line, 1);
fprintf(out, "%s %0.5d %0.5d 12 %d\n", name[0], nwfm[0], nwfm[1], qm[i]);
fprintf(out, "%s %0.5d %0.5d 13 %d\n", name[0], nwfm[1], nwfm[2], qm[i]);}
/* DPT1 */
else if (!(strcmp(l[i], "ay") || !(strcmp(l[i], "aw")) )
    {timeout(fm, nwfm, line, 1);
        fprintf(out, "%s %0.5d %0.5d 7 %d7\n", name[0], nwfm[0], nwfm[1], qm[i]);
        fprintf(out, "%s %0.5d %0.5d 23 %d7\n", name[0], nwfm[1], nwfm[2], qm[i]);}
/* DPT2 */
else if (!(strcmp(l[i], "ey") || !(strcmp(l[i], "oy")) )
    {timeout(fm, nwfm, line, 1);
        fprintf(out, "%s %0.5d %0.5d 31 %d7\n", name[0], nwfm[0], nwfm[1], qm[i]);
        fprintf(out, "%s %0.5d %0.5d 23 %d7\n", name[0], nwfm[1], nwfm[2], qm[i]);}
/* special case 'en' */
else if (!(strcmp(l[i], "en"))
    {timeout(fm, nwfm, line, 1);
        fprintf(out, "%s %0.5d %0.5d 31 %d7\n", name[0], nwfm[0], nwfm[1], qm[i]);
        fprintf(out, "%s %0.5d %0.5d 5 %d5\n", name[0], nwfm[1], nwfm[2], qm[i]);}

/* State2A, entered by + a release of a consonant */
/* after a . a consonant before another at beginning of syllable */
void state2A(FILE *out, int *fm, char l[MAXLEN][4], int line, char name[1][10], char *cue, int *qm)
{
    int i = line + 1;
    int nwfm[3];
    /* DELETION */
    if (!(strcmp(l[i], "xxx"))
        return;
    / * Find which consonant group the phonetics belong to */
    / * When found, can produce playlist line at the */
    / * side position */
    / * D GRP */
    if (!(strcmp(l[i], "d") || !(strcmp(l[i], "dcl") || !(strcmp(l[i], "dx")) || !(strcmp(l[i], "p") || !(strcmp(l[i], "pcl")) || !(strcmp(l[i], "zh")) || !(strcmp(l[i], "dd"))))
        fprintf(out, "%s %0.5d %0.5d 0 %d0\n", name[0], fm[line], fm[i+l], qm[i]);
    / * G GRP */
    else if (!(strcmp(l[i], "g") || !(strcmp(l[i], "gc1") || !(strcmp(l[i], "jh") || !(strcmp(l[i], "th")))))
        fprintf(out, "%s %0.5d %0.5d 1 %d1\n", name[0], fm[line], fm[i+l], qm[i]);
    / * H GRP */
    else if (!(strcmp(l[i], "h") || !(strcmp(l[i], "s") || !(strcmp(l[i], "r"))))
        fprintf(out, "%s %0.5d %0.5d 2 %d2\n", name[0], fm[line], fm[i+l], qm[i]);
    / * K GRP */
    else if (!(strcmp(l[i], "k") || !(strcmp(l[i], "kc1") || !(strcmp(l[i], "h") || !(strcmp(l[i], "v"))) || !(strcmp(l[i], "dh"))))
        fprintf(out, "%s %0.5d %0.5d 3 %d3\n", name[0], fm[line], fm[i+l], qm[i]);
    / * L GRP */
    else if (!(strcmp(l[i], "l") || !(strcmp(l[i], "sh") || !(strcmp(l[i], "w"))))
        fprintf(out, "%s %0.5d %0.5d 4 %d4\n", name[0], fm[line], fm[i+l], qm[i]);
    / * N GRP */
    else if (!(strcmp(l[i], "n") || !(strcmp(l[i], "b") || !(strcmp(l[i], "bc1"))))
        fprintf(out, "%s %0.5d %0.5d 5 %d5\n", name[0], fm[line], fm[i+l], qm[i]);
    / * NG GRP */
    else if (!(strcmp(l[i], "ng") || !(strcmp(l[i], "y") || !(strcmp(l[i], "ch")))))
        fprintf(out, "%s %0.5d %0.5d 6 %d6\n", name[0], fm[line], fm[i+l], qm[i]);
    / * T GRP */
}
else if (!(strcmp(l[i], "t"))) || !(strcmp(l[i], "cl")) || !(strcmp(l[i], "m")) ||
!(strcmp(l[i], "d")) /* CHIN */
else if (!(strcmp(l[i], "ao")) || !(strcmp(l[i], "eh")) ||
!(strcmp(l[i], "uw")) || !(strcmp(l[i], "ux"))) /* THROAT */
else if (!(strcmp(l[i], "ae")) || !(strcmp(l[i], "ih")) ||
!(strcmp(l[i], "ix")) || !(strcmp(l[i], "uh"))) /* MOUTH */
else if (!(strcmp(l[i], "aa")) || !(strcmp(l[i], "ow"))) /* SIED */
else if (!(strcmp(l[i], "ax"))) /* DPT1 */
else if (!(strcmp(l[i], "ay")) || !(strcmp(l[i], "aw"))) /* DPT2 */
else if (!(strcmp(l[i], "en"))) /* special case 'en' */
}

/* State3, entered by a release of a plosive after a beginning of a syllable */
void state3(FILE *out, int *fm, char l[MAXLEN][4], int line, char name[1][10], char *cue, int *qm)
{
int i = line + 1;
in
int nwfmi[3];
/* DELETION */
if (!(strcmp(l[i], "xxx")))
cue[i+1] = '*';
/* D GRP */
else if (!(strcmp(l[i], "d")) || !(strcmp(l[i], "cl")) || !(strcmp(l[i], "dx")) ||
!(strcmp(l[i], "p")) || !(strcmp(l[i], "pc1")) || !(strcmp(l[i], "zh")) ||
!(strcmp(l[i], "dd")))
state5(out, fm, i, line+2, 0, name, cue, qm);
/* G GRP */
else if (!(strcmp(l[i], "g")) || !(strcmp(l[i], "gc1")) ||
!(strcmp(l[i], "jh")) || !(strcmp(l[i], "ch")))
state5(out, fm, l, line+2, 1, name, cue, qm);
/* H GRP */
else if (!(strcmp(l[i], "h")) || !(strcmp(l[i], "s")) || !(strcmp(l[i], "r")))
  state5(out, fm, l, line+2, 2, name, cue, qm);
/* K GRP */
else if (!(strcmp(l[i], "k")) || !(strcmp(l[i], "kcl")) || !(strcmp(l[i], "v")) || !(strcmp(l[i], "dh")) || !(strcmp(l[i], "z")))
  state5(out, fm, l, line+2, 3, name, cue, qm);
/* L GRP */
else if (!(strcmp(l[i], "l")) || !(strcmp(l[i], "sh")) || !(strcmp(l[i], "w")))
  state5(out, fm, l, line+2, 4, name, cue, qm);
/* N GRP */
else if (!(strcmp(l[i], "n")) || !(strcmp(l[i], "b")) || !(strcmp(l[i], "bcl")))
  state5(out, fm, l, line+2, 5, name, cue, qm);
/* NG GRP */
else if (!(strcmp(l[i], "ng")) || !(strcmp(l[i], "y")) || !(strcmp(l[i], "ch")))
  state5(out, fm, l, line+2, 6, name, cue, qm);
/* T GRP */
else if (!(strcmp(l[i], "t")) || !(strcmp(l[i], "tcl")) || !(strcmp(l[i], "m")) || !(strcmp(l[i], "f")) || !(strcmp(l[i], "dt")))
  state5(out, fm, l, line+2, 7, name, cue, qm);
/* CHIN */
else if (!(strcmp(l[i], "ao1")) || !(strcmp(l[i], "eh")) || !(strcmp(l[i], "uw")) || !(strcmp(l[i], "ux")))
  fprintf(out, "%s %0.5d %0.5d 31 %d7\n", name[0], fm[line], fm[i+1], qm[i]);
  cue[i+1] = '*';
/* THROAT */
else if (!(strcmp(l[i], "ae")) || !(strcmp(l[i], "ih")) || !(strcmp(l[i], "ix")) || !(strcmp(l[i], "uh")))
  fprintf(out, "%s %0.5d %0.5d 23 %d7\n", name[0], fm[line], fm[i+1], qm[i]);
  cue[i+1] = '';
/* MOUTH */
else if (!(strcmp(l[i], "iy")) || !(strcmp(l[i], "er")) || !(strcmp(l[i], "axr")))
  fprintf(out, "%s %0.5d %0.5d 15 %d7\n", name[0], fm[line], fm[i+1], qm[i]);
  cue[i+1] = '*';
/* SIDE */
else if (!(strcmp(l[i], "aa")) || !(strcmp(l[i], "ow")))
  fprintf(out, "%s %0.5d %0.5d 7 %d7\n", name[0], fm[line], fm[i+1], qm[i]);
  cue[i+1] = '';}
/* SIDED */
else if (!(strcmp(l[i], "ah")) || !(strcmp(l[i], "ax")))
  timeout(fm, nwfm, i, 2);
  fprintf(out, "%s %0.5d %0.5d 7 %d7\n", name[0], nwfm[0], nwfm[1], qm[i]);
  fprintf(out, "%s %0.5d %0.5d 39 %d7\n", name[0], nwfm[1], nwfm[2], qm[i]);
  cue[i+1] = '*';
/* DPT1 */
else if (!(strcmp(l[i], "ay")) || !(strcmp(l[i], "aw")))
  timeout(fm, nwfm, i, 2);
  fprintf(out, "%s %0.5d %0.5d 7 %d7\n", name[0], nwfm[0], nwfm[1], qm[i]);
  fprintf(out, "%s %0.5d %0.5d 23 %d7\n", name[0], nwfm[1], nwfm[2], qm[i]);
  cue[i+1] = '*';
/* DPT2 */
else if (!(strcmp(l[i], "ey")) || !(strcmp(l[i], "oy")))
  timeout(fm, nwfm, i, 2);
fprintf(out, "%s %0.5d %0.5d 31 \d7\n", name[0], nwfm[0], nwfm[1], qm[i]);
fprintf(out, "%s %0.5d %0.5d 23 %d5\n", name[0], nwfm[1], nwfm[2], qm[i]);
cue[i+1] = '*';}

/* special case 'en' */
else if (!strchr(l[i],"en"))
    {timeout(fm, nwfm, i, 2);
     fprintf(out, "%s %0.5d %0.5d 31 %d7\n", name[0], nwfm[0], nwfm[1], qm[i]);
     fprintf(out, "%s %0.5d %0.5d 23 %d5\n", name[0], nwfm[1], nwfm[2], qm[i]);
     cue[i+1] = '*';}

/* State4, entered by , a vowel after a consonant */
void state4(FILE *out, int *fm, char l[MAXLEN][4], int line, int shp, char name[1][10], char *cue, int *qm)
{
    int i = line;
    int pos, pos2, shp2=8;
    int nwfm[3];
    if (qm[i-1] == 9 || qm[i] == 9)
        shp2 = 9;
    /* DELETION */
    if (!strstr(l[i],"xxx")) || (cue[i] != ','))
        fprintf(out, "%s %0.5d %0.5d %d %d%d\n", name[0], fm[i-1], fm[i], shp, shp2, shp);
    /* Find which vowel group the phonetic belongs to */
    /* When vowel group is found, use it in conjunction with the */
    /* hand shape passed in to produce playlist line */
    / * CHIN */
    else if (!strstr(l[i],"ao") || !strstr(l[i],"eh") ||
        !strstr(l[i],"uw") || !strstr(l[i],"ux"))
    {
    switch(shp)
    { case 0: pos = 24; break;
      case 1: pos = 25; break;
      case 2: pos = 26; break;
      case 3: pos = 27; break;
      case 4: pos = 28; break;
      case 5: pos = 29; break;
      case 6: pos = 30; break;
      case 7: pos = 31; break;
      default: break; }
    fprintf(out, "%s %0.5d %0.5d %d %d%d\n", name[0], fm[i-1], fm[i+1], pos, shp2, shp);
    }
    /* THROAT */
    else if (!strstr(l[i],"ae") || !strstr(l[i],"ih") ||
        !strstr(l[i],"ix") || !strstr(l[i],"uh"))
    {
    switch(shp)
    { case 0: pos = 16; break;
      case 1: pos = 17; break;
      case 2: pos = 18; break;
      case 3: pos = 19; break;
      case 4: pos = 20; break;
      case 5: pos = 21; break;
      case 6: pos = 22; break;
      case 7: pos = 23; break;
      default: break; }
    fprintf(out, "%s %0.5d %0.5d %d %d%d\n", name[0], fm[i-1], fm[i+1], pos, shp2, shp);
/ * MOUTH * /
else if (!(strcmp(l[i], "iy")) || !(strcmp(l[i], "er")) || !(strcmp(l[i], "axr")))
  {switch(shp)
    {case 0: pos = 8; break;
    case 1: pos = 9; break;
    case 2: pos = 10; break;
    case 3: pos = 11; break;
    case 4: pos = 12; break;
    case 5: pos = 13; break;
    case 6: pos = 14; break;
    case 7: pos = 15; break;
    default: break;
  }
  fprintf(out, "%s %0.5d %0.5d %d %d %d
", name[0], fm[i-1], fm[i+1], pos, shp2, shp);
}

/ * SIDE * /
else if (!(strcmp(l[i], "aa")) || !(strcmp(l[i], "ow")))
  {switch(shp)
    {case 0: pos = 0; break;
    case 1: pos = 1; break;
    case 2: pos = 2; break;
    case 3: pos = 3; break;
    case 4: pos = 4; break;
    case 5: pos = 5; break;
    case 6: pos = 6; break;
    case 7: pos = 7; break;
    default: break;
  }
  timeout(fin, nwfm, line, 2);
  fprintf(out, "%s %0.5d %0.5d %d %d %d
", name[0], nwfm[0], nwfm[1], pos, shp2, shp);
  fprintf(out, "%s %0.5d %0.5d %d %d %d
", name[0], nwfm[1], nwfm[2], pos2, shp2, shp);
}

/ * SIED * /
else if (!(strcmp(l[i], "ah")) || !(strcmp(l[i], "ax")))
  {switch(shp)
    {case 0: pos = 0; pos2 = 32; break;
    case 1: pos = 1; pos2 = 33; break;
    case 2: pos = 2; pos2 = 34; break;
    case 3: pos = 3; pos2 = 35; break;
    case 4: pos = 4; pos2 = 36; break;
    case 5: pos = 5; pos2 = 37; break;
    case 6: pos = 6; pos2 = 38; break;
    case 7: pos = 7; pos2 = 39; break;
    default: break;
  }
  timeout(fm, nwfm, line, 2);
  fprintf(out, "%s %0.5d %0.5d %d %d %d
", name[0], nwfm[0], nwfm[1], pos, shp2, shp);
  fprintf(out, "%s %0.5d %0.5d %d %d %d
", name[0], nwfm[1], nwfm[2], pos2, shp2, shp);
}

/ * DPT1 * /
else if (!(strcmp(l[i], "ay")) || !(strcmp(l[i], "aw")))
  {switch(shp)
    {case 0: pos = 0; break;
    case 1: pos = 1; break;
    case 2: pos = 2; break;
    case 3: pos = 3; break;
    case 4: pos = 4; break;
    case 5: pos = 5; break;
    default: break;
  }
case 6: pos = 6; break;
case 7: pos = 7; break;
default: break; }
timeout(fm, nwfm, line, 2);
fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], nwfm[0], nwfm[1], pos, shp2, shp);
fprintf(out, "%s %0.5d %0.5d 23 %d\n", name[0], nwfm[1], nwfm[2], shp2);
}
/* DPT2 */
else if (!(strcmp(l[i], "ey")) || !(strcmp(l[i], "oy")))
{
  switch(shp)
  {
    case 0: pos = 24; break;
    case 1: pos = 25; break;
    case 2: pos = 26; break;
    case 3: pos = 27; break;
    case 4: pos = 28; break;
    case 5: pos = 29; break;
    case 6: pos = 30; break;
    case 7: pos = 31; break;
    default: break;
  }
timeout(fm, nwfm, line, 2);
fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], nwfm[0], nwfm[1], pos, shp2, shp);
fprintf(out, "%s %0.5d %0.5d 23 %d\n", name[0], nwfm[1], nwfm[2], shp2);
}
/* special case 'en' */
else if (!(strcmp(l[i], "en")))
{
  switch(shp)
  {
    case 0: pos = 24; break;
    case 1: pos = 25; break;
    case 2: pos = 26; break;
    case 3: pos = 27; break;
    case 4: pos = 28; break;
    case 5: pos = 29; break;
    case 6: pos = 30; break;
    case 7: pos = 31; break;
    default: break;
  }
timeout(fm, nwfm, line, 2);
fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], nwfm[0], nwfm[1], pos, shp2, shp);
fprintf(out, "%s %0.5d %0.5d 23 %d\n", name[0], nwfm[1], nwfm[2], shp2);
}
/* Vowel -> Consonant */
else
{
fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], fm[i-1], fm[i], shp, qm[i-1], shp);
cue[i] = ' ';}
}
/* State5, entered by a vowel after a consonant */
void state5(FILE *out, int *fm, char l[MAXLEN][4], int line, int shp, char name[1][10], char *cue, int *qm)
{
int i = line;
int pos, pos2, shp2 = 8;
int nwfm[3];
if (qm[i-1] == 9 || qm[i] == 9)
  shp2 = 9;
/* DELETION */
if (!strcmp(l[i], "xxx")) || (cue[i] != ',')
    {fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], fm[i-2], fm[i], shp, shp2, shp);
     return;}

/* Find which vowel group the phonetic belongs to */
/* When vowel group is found, use it in conjunction with the */
/* hand shape passed in to produce playlist line */
/* CHIN */
else if (!strcmp(l[i], "ao") || !strcmp(l[i], "eh") || !strcmp(l[i], "uw") || !strcmp(l[i], "ux") {switch(shp)
    {case 0: pos := 24; break;
     case 1: pos = 25; break;
     case 2: pos = 26; break;
     case 3: pos = 27; break;
     case 4: pos = 28; break;
     case 5: pos = 29; break;
     case 6: pos = 30; break;
     case 7: pos = 31; break;
     default: break;
    }
    fprintf(out, "%s %0.5d %0.5d %d %d %d\n", name[0], fm[i-2], fm[i+1], pos, shp2, shp);
}
/* THROAT */
else if (!strcmp(l[i], "ae") || !strcmp(l[i], "ih") || !strcmp(l[i], "ix") || !strcmp(l[i], "uh") {switch(shp)
    {case 0: pos = 16; break;
     case 1: pos = 17; break;
     case 2: pos = 18; break;
     case 3: pos = 19; break;
     case 4: pos = 20; break;
     case 5: pos = 21; break;
     case 6: pos = 22; break;
     case 7: pos = 23; break;
     default: break;
    }
    fprintf(out, "%s %0.5d %0.5d %d %d %d\n", name[0], fm[i-2], fm[i+1], pos, shp2, shp);
}
/* MOUTH */
else if (!strcmp(l[i], "iy") || !strcmp(l[i], "er") || !strcmp(l[i], "axr") {switch(shp)
    {case 0: pos = 8; break;
     case 1: pos = 9; break;
     case 2: pos = 10; break;
     case 3: pos = 11; break;
     case 4: pos = 12; break;
     case 5: pos = 13; break;
     case 6: pos = 14; break;
     case 7: pos = 15; break;
     default: break;
    }
    fprintf(out, "%s %0.5d %0.5d %d %d\n", name[0], fm[i-2], fm[i+1], pos, shp2, shp);
}
/* SIDE */
else if (!strcmp(l[i], "aa") || !strcmp(l[i], "ow") {switch(shp)
    {case 0: pos = 0; break;
     
}
case 1: pos = 1; break;
case 2: pos = 2; break;
case 3: pos = 3; break;
case 4: pos = 4; break;
case 5: pos = 5; break;
case 6: pos = 6; break;
case 7: pos = 7; break;
default: break; }

fprintf(out, "%s %.5d %.5d %.d\n", name[0], fm[i-2], fm[i+1], pos, shp2, shp); }

/* S 1 ED */
else if (!(strcmp(l[i],"ah")) || !(strcmp(l[i],"ax")))
{switch(shp)
  {case 0: pos = 0; pos2 = 32; break;
case 1: pos = 1; pos2 = 33; break;
case 2: pos = 2; pos2 = 34; break;
case 3: pos = 3; pos2 = 35; break;
case 4: pos = 4; pos2 = 36; break;
case 5: pos = 5; pos2 = 37; break;
case 6: pos = 6; pos2 = 38; break;
case 7: pos = 7; pos2 = 39; break;
default: break; }

timeout(fm, nwfm, line, 3);
fprintf(out, "%s %.5d %.5d %.d\n", name[0], nwfm[0], nwfm[1], pos, shp2, shp); }

/* DPT 1 */
else if (!(strcmp(l[i],"ay")) || !(strcmp(l[i],"aw"))
{switch(shp)
  {case 0: pos = 0; break;
case 1: pos = 1; break;
case 2: pos = 2; break;
case 3: pos = 3; break;
case 4: pos = 4; break;
case 5: pos = 5; break;
case 6: pos = 6; break;
case 7: pos = 7; break;
default: break; }

timeout(fm, nwfm, line, 3);
fprintf(out, "%s %.5d %.5d %.d\n", name[0], nwfm[1], nwfm[2], pos2, shp2, shp); }

/* DPT 2 */
else if (!(strcmp(l[i],"ey")) || !(strcmp(l[i],"oy")))
{switch(shp)
  {case 0: pos = 24; break;
case 1: pos = 25; break;
case 2: pos = 26; break;
case 3: pos = 27; break;
case 4: pos = 28; break;
case 5: pos = 29; break;
case 6: pos = 30; break;
case 7: pos = 31; break;
default: break; }
timeout(fm, nwfm, line, 3);
fprintf(out, "%s %0.5d %0.5d %d %d
", name[0], nwfm[0], nwfm[1], pos, shp2, shp);
fprintf(out, "%s %0.5d %0.5d 23 %d7
", name[0], nwfm[1], nwfm[2], shp2);
}
/* special case 'en' */
else if (!(strcmp(l[i],"en")))
{
switch(shp)
{case 0: pos = 24; break;
  case 1: pos = 25; break;
  case 2: pos = 26; break;
  case 3: pos = 27; break;
  case 4: pos = 28; break;
  case 5: pos = 29; break;
  case 6: pos = 30; break;
  case 7: pos = 31; break;
  default: break; }
timeout(fm, nwfm, line, 3);
fprintf(out, "%s %0.5d %0.5d %d %d
", name[0], nwfm[0], nwfm[1], pos, shp2, shp);
fprintf(out, "%s %0.5d %0.5d 5 %d5
", name[0], nwfm[1], nwfm[2], shp2);
}
/* Vowel -> Consonant */
else
{
fprintf(out, "%s %0.5d %0.5d %d %d
", name[0], nwfm[0], nwfm[1], pos, shp2, shp);
cue[i] = '*' ;
}
}

void timeout(int *tm, int *frm, int line, int opt)
{
  int temp;
/* return 1 line time */
  if (opt == 1)
    {frm[0] = tm[line];
     temp = (tm[line+1] - tm[line]) / 2;
     frm[1] = tm[line] + temp;
     frm[2] = tm[line+1];}
  if (opt == 2)
    {frm[0] = tm[line-1];
     temp = (tm[line+1] - tm[line-1]) / 2;
     frm[1] = tm[line-1] + temp;
     frm[2] = tm[line+1];}
  if (opt == 3)
    {frm[0] = tm[line-2];
     temp = (tm[line+1] - tm[line-2]) / 2;
     frm[2] = tm[line+1];}
}

int frameout(double *tm, int *frm, int start, int stop)
{
  int i = 0;
  while (tm[i] != '\0')
  { tm[i] = tm[i] - 530;
    tm[i] = tm[i] * f;
    frm[i] = ((int) (0.5 + tm[i]/1000*30)) + start;
}
if ((frm[0] < start) || (frm[i-1] > stop))
    {printf("ERROR: OVERWRITING BLUE\n");
     return(0);}
return(1);
}

/****************************************************************************
* Function errmark marks half the error phones (10\% total phones) and 10\% of the nonerror phones. 9 means that the cue should receive a red box outline and 8 means that the cue should remain the same. The phones that received marks were decided with a random number generator.*/
/****************************************************************************

void errmark(char err[MAXLEN][5], int *qm, int n)
{
    float weper = .5;
    int nerr[MAXLEN];
    int werr[MAXLEN];
    int i,j=0,k=0, ntot=0, wtot=0, nemark, wemark;
    short rand1, rand3;
    double rand2;
    srand(time(0));

    /* divide phone array into 2 array, one of errors, one nonerrors */
    for (i = 0; strcmp(err[i], "") != 0; i++)
        {if (err[i][1] == '*')
            {nerr[i] = 0; werr[i] = 8; wtot = wtot + 1;}
        else
            {nerr[i] = 8; werr[i] = 0; ntot = ntot + 1;}
    }

    /* figure out how many phones should receive marks */
    wemark = weper * wtot; nemark = wemark;
    /* assign marks to the nonerrors */
    while (j < nemark)
        {rand1 = rand(); rand2 = abs(rand1);
         rand2 = rand2 / 32768 * n;
         rand3 = floor(rand2);
         if (nerr[rand3] == 8)
             {nerr[rand3] = 9; j++;}
    }

    /* assign marks to the errors */
    while (k < wemark)
        {rand1 = rand(); rand2 = abs(rand1);
         rand2 = rand2 / 32768 * n;
         rand3 = floor(rand2);
         if (werr[rand3] == 8)
             {werr[rand3] = 9; k++;}
    }

    /* combine the 2 arrays back into a single array */
    for (i = 0; i < n; i++)
        {if (nerr[i] == 0)
The program stateins.c is a short program that adds insertions at points specified by *.lgp files. The output of stateins.c is a transcription file that must then be processed by state.c to produce a playlist.

stateins.c

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <math.h>

#define MAXLEN 80 /* maximum number of lines in input file */

void gettime(char ph[MAXLEN][4], char avphon[MAXLEN][4], double *, int, double *);
void getphon(char avphon[MAXLEN][4], char newphon2[MAXLEN][4], int);

main(int argc, char *argv[])
{ FILE *sent_file; /* pointer to file to read in */
 FILE *out_file; /* pointer to file to put output */
 FILE *len_file; /* pointer to file that has phoneme lengths */
 char filename[30]; /* name of file to read in */
 char newphon[MAXLEN][4]; /* 2D array to hold phonemes */
 char newphon2[MAXLEN+1][4];
 double ftm[MAXLEN]; /* array to hold time values */
 double nftm[MAXLEN+1];
 char cue[MAXLEN]; /* array to hold cue markings */
 char ncue[MAXLEN+1];
 int i=0, j=0, k=0, n:L, n2, l=0; /* array index */
 char err[MAXLEN][5]; /* array to hold error markings */
 char oldphon[MAXLEN][4]; /* array to hold correct phonemes */
 char line[20]; /* array to hold each line of file */
 char avphon[MAXLEN][4];
 double avlen[MAXLEN];
 int count=1;

/* Get name of file from user */
if (argc == 1)
{ printf("\nInput name of sentence file> ");

qm[i] = werr[i];
    if (werr[i] == 0)
        qm[i] = nerr[i];
    }
qm[n] = NULL;
}
scanf("%s", filename);
sent_file = fopen(filename, "r");
out_file = stdout; }
else if (argc == 2)
{ sent_file = fopen(argv[1], "r");
out_file = stdout; }
else if (argc == 3)
{ sent_file = fopen(argv[1], "r");
out_file = fopen(argv[2], "w");
}

/* Divide file components (times, cues, phonemes) into separate arrays */
while (fgets(line, 40, sent_file) != NULL)
{ sscanf(line, "%lf/%c%3s %s %s", &ftm[i], &cue[i], newphon[i], oldphon[i], err[i]);
  sscanf(line, "%lf/%c%3s %*s %*s", &nftm[i], &ncue[i], newphon2[i]);
i++;
}
fclose(sent_file);

/* get a list of phones to use as insertions and their durations */
len_file = fopen("length.txt", "r");
while (fscanf(len_file, "%s %lf", avphon[k], &avlen[k]) != EOF)
  k++;
fclose(len_file);

while (nftm[l] != '\0')
  /* if the insertion symbol, make room for and get insert phone */
  { if (err[j][1] == '*' && !(strcmp(newphon[j], "ooo")))
      { j++;
        for (n1 = j+count; n1 <= i+count; n1++)
          { nftm[n1] = ftm[n2];
            ncue[n1] = cue[n2];
            newphon2[n1][0] = newphon[n2][0];
            newphon2[n1][1] = newphon[n2][1];
            newphon2[n1][2] = newphon[n2][2];
            newphon2[n1][3] = newphon[n2][3];
            n2++;
          } newphon2[j+count-1][0] = oldphon[j][0];
          newphon2[j+count-1][1] = oldphon[j][1];
          newphon2[j+count-1][2] = oldphon[j][2];
          newphon2[j+count-1][3] = oldphon[j][3];
          getphon(avphon, newphon2, j+count);
          gettime(newphon2, avphon, avlen, j+count, nftm);
          count = count+1;
        j++;
      }
    }
while (nftm[l] != '\0')
  { printf(out_file, "%lf/%c%s\n", nftm[l], ncue[l], newphon2[l]);
    l++;
  }
fclose(out_file);

/* Function getphon randomly selects a phone to be inserted */
void getphon(char list[MAXLEN][4], char phon[MAXLEN][4], int i)
{ short rand1, j;
double rand2;

srand(time(0)+i);

rand1 = rand(); rand2 = abs(rand1);
rand2 = rand2 / 32768 * 43;
j = floor(rand2);
phon[i][0] = list[j][0];
phon[i][1] = list[j][1];
phon[i][2] = list[j][2];
phon[i][3] = list[j][3];
}

/* Function gettime adjusts the time of the inserted phone and its */
/* surrounding phones. This function tries not to allow any phone */
/* to have a duration shorter than 20ms. */
void gettime(char phon[MAXLEN][4], char plist[MAXLEN][4], double *llist, int n, double *ftm)
{
    double inslen, pre, post, fracl, frac2;
    int i=0;

    while(strcmp(phon[n],plist[i]))
    {
        inslen = llist[i];
        
        if (!strcmp(phon[n+1], ""))
        {
            pre = ftm[n+1] - ftm[n];
            post = 0;
        }
        else
        {
            pre = ftm[n+2] - ftm[n+1];
        }
        /* while the inserted phoneme is longer than sum of surrounding phonemes */
        while (inslen > pre + post || (inslen > pre && inslen > post))
        {
            inslen = inslen * .8;
        }
        if (pre < 20) /* if the pre phoneme is less than 20ms */
        {
            while (inslen > post) /* insert into post phoneme */
            {
                inslen = inslen * .8;
                ftm[n] = ftm[n+1];
                ftm[n+1] = ftm[n] + inslen;
            }
        }
        else if (post < 20) /* if the post phoneme is less than 20ms */
        {
            while (inslen > pre) /* insert into pre phoneme */
            {
                inslen = inslen * .8;
                ftm[n] = ftm[n+1] - inslen;
            }
        }
        else
        {
            frac1 = pre / (pre + post);
            frac2 = post / (pre + post);
            ftm[n] = ftm[n+1] - (inslen * frac1);
            ftm[n+1] = ftm[n+1] + (inslen * frac2);
        }
    }
}

The new version of State converts output from the HTK recognizer into playlists without requiring syllable symbols (instead the program figures them out). Listed below is the main body of newstate.c and the one function not included in state.c. The other functions for newstate.c are the same as state.c.
newstate.c

/* This is the newest version of State that does not require the */
/* syllable symbols. */
/* This program converts the sentence file output of the HTK */
/* recognizer into playlists that can be used to superimpose cues */
/* on prerecorded sentences on videodisc. */
/* Before the playlist is complete, a POSITION file and a SHAPE file */
/* needs to be appended to the beginning of the output file. */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXLEN 80 /* maximum number of lines in input file */
#define .96432 /* correction factor for sampling */

void state1(FILE *, int *, char ph[MAXLEN][4], int, char sent[1][10], char *);
void state2(FILE *, int *, char ph[MAXLEN][4], int, char sent[1][10], char *);
void state2A(FILE *, int *, char ph[MAXLEN][4], int, char sent[1][10], char *);
void state3(FILE *, int *, char ph[MAXLEN][4], int, char sent[1][10], char *);
void state4(FILE *, int *, char ph[MAXLEN][4], int, char sent[1][10], char *);
void state5(FILE *, int *, char ph[MAXLEN][4], int, int, char sent[1][10], char *);
void timeout(int *, int *, int, int);
int frameout(double , int, int *, int, int);
void getphon(char ph[MAXLEN][4]);

main(int argc, char *argv[])
{
    FILE *sent_file; /* pointer to file to read in */
    FILE *frame_file; /* pointer to file that has frame info */
    FILE *out_file; /* pointer to file to put output */
    char filename[30]; /* name of file to read in */
    char phon[MAXLEN][4]; /* 2D array to hold phonemes lineXphon. length */
    char sent[1][10]; /* array for sentence id */
    int startfm, stopfm; /* beginning frame and end frame (nonblue) */
    double ftn1[MAXLEN]; /* array to hold time values */
    double ftn2[MAXLEN];
    char cue[MAXLEN]; /* array to hold cue markings */
    int frame[MAXLEN]; /* array to hold absolute frames numbers */
    int i = 0, j, k; /* array index */
    char line[50];
    char ph1, ph2;

    /* Get name of file from user */
    if (argc == 1)
    {
        printf("\nInput name of sentence file> ");
        scanf("%s", filename);
        sent_file = fopen(filename, "r");
        out_file = stdout;
    }
    else if (argc == 2)
    {
        sent_file = fopen(argv[1], "r");
        out_file = stdout;
    }
    else if (argc == 3)
sentfile = fopen(argv[1], "r");
outfile = fopen(argv[2], "w");

/* Divide file components (times, phonemes) into separate arrays */
while (fgets(line, 50, sentfile) != NULL)
    {sscanf(line, "%lf %lf %s", &ftml[i], &ftm2[i], phon[i]);
        i++;
    }
fclose(sent_file);

/* extract phon from recognizer file */
getphon(phon);

/* Reads file that contains information on sentence */
/* and start and stop frames for the sentence (nonblue) */
frame_file = fopen("finfo.dat", "r");
if (fscanf(frame_file, "%s%d%d", sent[0], &startfm, &stopfm) != EOF)
    ;
fclose(frame_file);

/* Convert times to frame values. Exit if overwriting into blue. */
if ((frameout(ftml, ftm2[i-1], frame, startfm, stopfm)) == 0)
    printf("IN SENTENCE %s
", sent[0]);

fprintf(out_file, "\nSENTENCES\n{\n" );
fprintf(out_file, "%s %.0.5d %0.5d\n\}\n", sent[0], startfm-2, stopfm+3);
fprintf(out_file, "\nPLAYLIST\n{\n" );

/* Implement the state diagram */
for (j=0; frame[j] != '\0'; j++)
    {ph1 = phon[j][0];
       ph2 = phon[j+1][0];
       if (ph1 == ' ')
           {cue[j] = '"';
            cue[j+1] = ',';
            j = j + 1;
        }
        else if (ph1 == 'b' || ph1 == 'c' || ph1 == 'd' || ph1 == 'f' || ph1 == 'g' || ph1 == 'h' || ph1 == 'j')
            {if (ph1 == 'b' || ph1 == 'c' || ph1 == 'd' || ph1 == 'f' || ph1 == 'g' || ph1 == 'h' || ph1 == 'j')
                 {if (ph2 == 'a' || ph2 == 'e' || ph2 == 'i' || ph2 == 'o' || ph2 == 'u')
                      {cue[j] = '"';
                       cue[j+1] = ',';
                       j = j + 1;
                   }
                   else if (!strcmp(phon[j], "vcl"))
                       {if (phon[j+2][0] == 'a' || phon[j+2][0] == 'e' || phon[j+2][0] == 'i' || phon[j+2][0] == 'o' || phon[j+2][0] == 'u')
                           {cue[j] = ' ';
                            cue[j+1] = '+';
                            cue[j+2] = '"';
                            j = j + 2;
                        }
                        else
                            {cue[j] = ' '*;
                             cue[j+1] = '+';
                             j = j + 1;
                         }
                     }
                     else
                         {cue[j] = ' '*;
                         j = j + 1;
                     }
                 }
             }
         }
     }
cue[j] = " ";

for (k=0; frame[k] != '\0'; k++)
{
    switch (cue[k])
    {
    case '"':
        break;
    case ':'
        if (cue[k+1] == '+')
            state3(outfile,frame,phon,k,sent,cue);
        else
            state1(outfile,frame,phon,k,sent,cue);
        break;
    case '*':
        if (cue[k+1] == '+')
            state2A(outfile,frame,phon,k,sent,cue);
        else
            state2(outfile,frame,phon,k,sent,cue);
        break;
    case ',':
        break;
    case '+':
        break;
    default:
        break;
    }
}
printf(outfile, "\n");
fclose(outfile);
}

/* Function getphon extracts the phones from the recognizer sentence files */
void getphon(char phon[MAXLEN][4])
{
    int i=0, j;
    char temp[MAXLEN][4];

    while (strcmp(phon[i],""))
    {
        if (i ==0)
            ;
        else { for (j=0; phon[i][j] != '+'; j++)
                    phon[i-1][j] = phon[i][j];
                    phon[i-1][j] = '\0';
                i++;}
        phon[i-1][0] = '"';
        phon[i-1][1] = '\0';
    }

Finally, the program errormat.c is included because it was used to count the number
of phone errors and to determine which errors caused cue errors.

errormat.c

/*****************************/
This program takes an *.lgp program and calculates four numbers:

1). Total number of phones in sentence
2). Total number of error phones
3). Number of error phones that caused cue changes
4). Number of error phones that did not cause cue changes

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <math.h>
#define ML 80 /* maximum number of lines in input file */

void init(int mat[ML][ML]);
void compare(char oldphon[ML][4], char newphon[ML][4], int, int mat[ML][ML], char avphon[ML][4], int *);

main(int argc, char *argv[])
{
 FILE *sent_file; /* pointer to file to read in */
 FILE *out_file; /* pointer to file to put output */
 FILE *phon_file; /* pointer to file that has phonemes */
 char filename[30]; /* name of file to read in */
 double ftm[ML];
 char cue[ML];
 char newphon[ML][4]; /* 2D array to hold phonemes lineXphon. length */
 int i=0, j=0, k=0, l=0; /* array index */
 char err[ML][5]; /* array to hold error markings */
 char oldphon[ML][4]; /* array to hold correct phonemes */
 char line[20]; /* array to hold each line of file */
 char avphon[ML][4];
 int mat[ML][ML];
 int count = 0;
 int errors = 0;
 int newcue[2];

 /* Get name of file from user */
 if (argc == 1)
 {
 printf("Input name of sentence file> ");
 scanf("%s",filename);
 sent_file = fopen(filename, "r");
 out_file = stdout;
 }
 else if (argc == 2)
 {
 sent_file = fopen(argv[1], "r");
 out_file = stdout;
 }
 else if (argc == 3)
 {
 sent_file = fopen(argv[1], "r");
 out_file = fopen(argv[2], "w");
 }
 /* Divide file components (times, cues, phonemes) into separate arrays */
 while (fgets(line,40,sent_file) != NULL)
 {
 sscanf(line, "%lf/%c%3s %s %s %s", &ftm[i], &cue[i], newphon[i], oldphon[i], err[i]);
 if (!(strcmp(oldphon[i],"bc1") || !(strcmp(oldphon[i],"dcl") || !(strcmp(oldphon[i],"gcl") || !(strcmp(oldphon[i],"kc1")))))
     count = count + 1;
     errors = errors + 1;
     newcue[j] = i;
     j = j + 1;
     if (j >= 2)
     {
     for (k = 0; k < ML; k++)
     {
     for (l = 0; l < ML; l++)
     {
     mat[i][l] = oldphon[j][k];
     }
     }
     }
     if (j == 0)
     {
     mat[i][0] = oldphon[0][0];
     }
     if (j == 1)
     {
     for (k = 0; k < ML; k++)
     {
     mat[i][k] = oldphon[1][k];
     }
     }
     if (j == 2)
     {
     for (k = 0; k < ML; k++)
     {
     mat[i][k] = oldphon[2][k];
     }
     }
!(strcmp(oldphon[i], "tcl")) || !(strcmp(oldphon[i], "pcl"))

else count = count + 1;
i++;

fclose(sent_file);

count = count - 1;

phon_file = fopen("phones.txt", "r");
while (fscanf(phon_file, "%s", avphon[k]) != EOF)
k++;
fclose(phon_file);

init(mat);
newcue[0] = 0; newcue[1] = 0;

while (cue[j] != ' ~')
    {if (errlj[j][1] == ' * ')
        {compare(oldphon, newphon, j, mat, avphon, newcue);
         errors = errors + 1;}
        j++;
    }

fprintf(out_file, "%s %d %d %d %d\n", argv[1], count, errors, newcue[0], newcue[1]);

/* This section that is commented out outputs a confusion matrix */
/* for (n=0; n < k; n++) */
/* {if (n == 0) */
/* {fprintf(outfile, " "); */
/* for (m=0; m < k; m++) */
/* fprintf(outfile, "%s ", avphon[m]); */
/* fprintf(outfile, "\n");} */
/* for (m=0; m < k; m++) */
/* {if (m == 0) */
/* fprintf(outfile, "%s ", avphon[m]); */
/* fprintf(outfile, "\n");} */
/* fprintf(outfile, "\n");} */

fclose(out_file);

void init(int mat[ML][ML])
{ int i, j;

    for (i=0; i < ML; i++)
        for (j=0; j < ML; j++)
            mat[i][j] = 0;

    /* This function determines if a cue change occurred or not */
void compare(char old[ML][4], char new[ML][4], int n, int mat[ML][ML], char plist[ML][4], int *newcue)
{ int i=0, j=0;
while(strcmp(old[n], plist[i]))
i++;
while(strcmp(new[n], plist[j]))
    j++;

if (i<=2 && j<=2) newcue[0] = newcue[0] + 1;
else if (i>2 && i<=5 && j>2 && j<=5) newcue[0] = newcue[0] + 1;
else if (i>5 && i<=7 && j>5 && j<=7) newcue[0] = newcue[0] + 1;
else if (i>7 && i<=9 && j>7 && j<=9) newcue[0] = newcue[0] + 1;
else if (i>9 && i<=11 && j>9 && j<=11) newcue[0] = newcue[0] + 1;
else if (i>11 && i<=13 && j>11 && j<=13) newcue[0] = newcue[0] + 1;
else if (i>13 && i<=15 && j>13 && j<=15) newcue[0] = newcue[0] + 1;
else if (i==16 && j==16) newcue[0] = newcue[0] + 1;
else if (i>16 && i<=20 && j>16 && j<=20) newcue[0] = newcue[0] + 1;
else if (i>20 && i<=25 && j>20 && j<=25) newcue[0] = newcue[0] + 1;
else if (i>25 && i<=30 && j>25 && j<=30) newcue[0] = newcue[0] + 1;
else if (i>30 && i<=35 && j>30 && j<=35) newcue[0] = newcue[0] + 1;
else if (i>35 && i<=40 && j>35 && j<=40) newcue[0] = newcue[0] + 1;
else if (i>40 && i<=45 && j>40 && j<=45) newcue[0] = newcue[0] + 1;
else if (i>45 && i<=50 && j>45 && j<=50) newcue[0] = newcue[0] + 1;

mat[j][i] = mat[j][i] + 1;