Infrastructure Development for Integration of Lip Reading into the SUMMIT Speech Recognizer

by

Chia-Hao La

S.B., Massachusetts Institute of Technology, 2002

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 and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2003

Author ....
Department of
Electrical Engineering and Computer Science
May 21, 2003

Certified by ....
Timothy J. Hazen
Research Scientist
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Infrastructure Development for Integration of Lip Reading into the SUMMIT Speech Recognizer

by

Chia-Hao La

Submitted to the Department of
Electrical Engineering and Computer Science
on May 21, 2003, in partial fulfillment of the
requirements for the degree of
Master of Engineering in
Electrical Engineering and Computer Science

Abstract

This thesis describes a method for augmenting an audio-only speech recognizer with visual lip-reading information, in order to improve the performance and robustness of the recognizer. The speech recognizer's variable length audio segments are resolved with the fixed length video frames using segment constrained Hidden Markov Modeling. A Viterbi search over the per-segment Hidden Markov Model resolves the variable asynchrony between the audio and video streams. The two streams are combined according to a relative weighting scheme, which is determined by optimizing on a held-out data set. Although a full audio-visual system has yet not been implemented, this thesis describes the infrastructure that has been developed to accommodate integration with a visual lip-reading module that will be completed in the near future.

Thesis Supervisor: Timothy J. Hazen
Title: Research Scientist

Acknowledgments

I would like to extend my sincerest gratitude to my supervisor T.J. Hazen, who has been an amazing mentor for the past year and a half. The insight, care, and good nature that he brings to everything he does is inspiring, and I know he will be a great professor if he ever decides to pursue teaching. He is one of the few people I know who truly loves sharing his knowledge and experience with others.

Next, I'd like to thank Teresa, whose constant warmth, humor, and support have always managed to keep me sane and optimistic. I'm also grateful to the grad students in SLS, especially my officemates Ed, Brooke, and Kate, for creating a great atmosphere to work in. Ed deserves special mention for being a great friend and a wonderful cook/musician to boot.

I'm appreciative of my aparment mates Rob Chang and Ed Huang, who I've known since freshman year. We have suffered through three class projects and many housing ordeals together. Their presence has made these potentially painful experiences easy to endure and even fun.

Finally, I am most indebted to my mother, father, and aunt. They have provided me with guidance and support in all aspects of life, and have taught me that being a good person is the most important form of success.

This research was supported by DARPA under contract N66001-99-1-8904 monitored through Naval Command, Control and Ocean Surveillance Center, and by an industrial consortium supporting the MIT Oxygen Alliance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Traditional speech recognition systems have relied purely on sound for input. One way to improve their performance is to augment the audio data with visual lip-reading information. By resolving the data received from both the audio and visual channels, one can achieve lower word error rates than by using either channel on its own. This technique is especially effective in noisy environments, where the audio waveform may be partially corrupted.

This thesis describes modifications to the SUMMIT speech recognizer [3] that are intended to facilitate integration with a visual channel. Although the full audio-visual speech recognizer has yet to be completed, most of the key integration issues have already been addressed. The following sections elaborate on the motivation for this project, as well the main issues one must account for during integration.

1.1 Motivation: Bimodality of Speech

When humans listen to a speaker, they often rely not only on hearing but also on sight. Observing the motions made by the speaker's lips and jaw enhances the intelligibility of spoken language. Figure 1-1 shows the performance of human listeners under various noise conditions, using both audio-only observations and audio-visual observations. The figure suggests that the visual modality provides only modest improvements in clear acoustic conditions, but becomes more significant in the presence

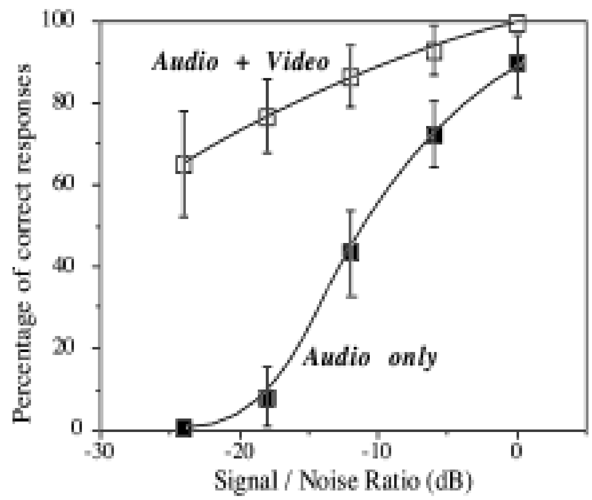

Figure 1-1: Improved recognition through vision of the speaker's face [1]

of noise [1].

It is also known that the vision and speech modalities complement each other. Many sounds that are confusable by ear are easily distinguishable by eye, such as *n* and *m*. Similarly, many phones that are difficult to distinguish by eye are easy to identify by ear, such as *p*, *b*, and *m*.

These results all point to the potential benefit of integrating the two modalities into a speech recognizer. If done successfully, the recognizer should be able to achieve at least modest improvements in normal acoustic environments, and substantial improvements in noisy conditions.

1.1.1 Phones and Visemes

Phonemes are the smallest linguistically distinguishable units of sound in the English language, and phones are the acoustic realizations of phonemes. Some examples of phones were mentioned above, such as *p*, *b*, and *m*. See Appendix A for a listing of the English phones, along with their IPA symbols and corresponding TIMIT labels,

Visemes are a set of visual units that correspond roughly to phones, and describe the visual realization of phonemes. In general, one can only see a speaker's lips and jaws, while the other articulators (e.g., the tongue and the glottis) are typically hidden from sight. Therefore, some visemes can actually correspond to more than one phone, resulting in a one-to-many mapping. For example, the phones b and p differ from each other only in that b is voiced. Since voicing occurs at the glottis, which is not typically visible, these two phones are visually indistinguishable and actually correspond to the same viseme.

1.2 Issues in Audio-Visual Integration

This section describes three issues that must be considered in combining any audio-visual speech recognizer. The issues are introduced, and several alternative solutions are proposed.

1.2.1 Early vs. Late Integration

One of the key questions to ask when integrating audio and visual data is "When should the information be combined?" In early integration, feature vectors from both modalities are concatenated into one large vector. This resulting vector is then processed by a joint audio-visual classifier, which uses the combined information to assign likelihoods to the recognizer's phonetic hypotheses.

In late integration, the audio data and video data are analyzed by separate classifiers. Each classifier processes its own data stream, and the two sets of outputs are combined in a later stage to produce the final hypothesis. The combination can occur at different levels. For example, visual visemes could be resolved against audio phones. Other possibilities include combination at the state, syllable, word, and utterance level.

1.2.2 Channel Weighting

Another issue to consider is the relative weighting of the two channels. Typically, more information can be extracted from the audio stream than the video stream. This is shown by the higher success rate of audio-only recognizers when compared to video-only recognizers [2]. These results are intuitive, since there are more phones than visemes, indicating that audio information provides more distinguishing cues. Thus, one would expect the audio channel to have higher priority over the video channel. This strategy is often accomplished by optimizing the relative weights on a held-out data set.

However, there are other factors to consider. For example, there might be substantial noise in one of the channels, in which case an integrity measure like the signal to noise ratio would be useful. Also, certain phones may be more reliably represented in one channel than the other. Thus, the weighting could be varied according to the specific sound being processed at any moment.

1.2.3 Modeling Asynchrony

There is an inherent asynchrony between the visual and audio cues of speech. Some phones may be pronounced roughly when the mouth moves to make the appropriate sound. However, sometimes the mouth prepares to form the sound hundreds of milliseconds before the phone is actually pronounced [1]. Thus, there must be some way to handle this variable asynchrony between the two channels. Some integration strategies handle it implicitly, for example by using early integration and training on a large amount of varied speech. Other strategies model the asynchronous properties explicitly, by using models that describe how much the two streams are offset from each other at any given time. Examples of both types of asynchrony resolution are discussed in greater detail in Section 2.2.

1.3 Goals

The SUMMIT speech recognizer [3] provides a front-end for conversational systems developed at the Spoken Language Systems Group at MIT's Laboratory for Computer Science. The eventual goal of this project is to improve SUMMIT's word error rate via integration with a visual lip-reading module. However, the production of a full audio-visual speech recognizer is a multi-person effort. This project attempts to provide the infrastructure necessary to perform this integration. The infrastructure consists of a method for resolving visual frame-level scores with audio segments, accounting for possible asynchrony between the two streams.

In the absence of a visual phone classifier, the infrastructure is evaluated using audio input for both streams. The purpose of the evaluation is to ensure that the techniques employed provide reasonable accuracy and speed, and will indeed be a useful platform on which to perform integration.

1.4 Outline

The rest of this paper is organized as follows. Chapter 2 introduces the modeling and search techniques that will be used in the thesis. It also discusses some previous work that has been done in audio-visual speech recognition, to provide suitable context for this project. Chapter 3 describes how the integration was performed in the SUMMIT recognizer. Chapter 4 explains how the infrastructure was tested and lists some results. Chapter 5 explains how an audio-visual database was assembled in preparation for integration. Finally, Chapter 6 provides some concluding remarks and suggests directions for future research.

Chapter 2

Background

This chapter describes standard speech recognition techniques used in this thesis, as well as previous work that has been done in the field.

2.1 Speech Recognition Techniques

In this section, Hidden Markov Models are introduced in a high level manner. Because they play a key role in the asynchrony modeling, it is important to understand their capabilities and operation. After that, an efficient way of computing the best path through a Hidden Markov Model given an observation sequence is described.

2.1.1 Hidden Markov Models

Hidden Markov Models (HMMs) [9] are a common way of modeling temporal events, such as the generation of a phone. The event's progression is represented by a traversal through a sequence of states, each of which generates an output. In the context of speech recognition, the output may be some kind of feature vector such as a set of Mel Frequency Cepstral Coefficients (MFCCs).

Figure 2.1.1 provides an example of a simple four-state HMM. The states are labeled 0 through 3, and the arcs represent possible transitions between states.

The HMM is very powerful because of its *probabilistic* nature. Any given state

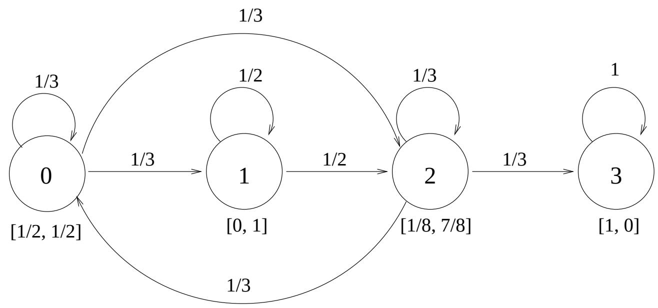

Figure 2-1: A four-state HMM

and observation sequence are generated by a given HMM with a certain probability. In general, the probability of emitting a certain observation from an HMM state is defined by a continuous density function (CDF) for each state. However, for the purposes of illustration the sample HMM discussed will use a discrete probability distribution function (PDF) with two possible observation values, namely 0 or 1. The probabilistic structure of any HMM can be defined by three sets of distributions. First, the initial state matrix π has elements π_i, and defines the probability of starting out in state i. Second, the transition matrix A has elements $a_{i,j}$, and yields the probability of moving from state i to state j at any time. Finally, the observation matrix B with elements $b_{i,k}$ defines how likely it is for state i to generate the k^{th} observation.

For example, the probability matrices for the HMM in Figure 2.1.1 might be as shown in Equation 2.1. Notice that the initial probability matrix π forces any state sequence to begin at state 0. The transition matrix A has non-zero elements corresponding to the transition arcs shown in Figure 2.1.1, and allows backward transitions since it is not upper-triangular.

$$\pi = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \quad A = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 \\ 0 & 1/2 & 1/2 & 0 \\ 1/3 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1/2 & 1/2 \\ 0 & 1 \\ 1/8 & 7/8 \\ 1 & 0 \end{bmatrix} \tag{2.1}$$

2.1.2 Hidden Markov Models in Speech Recognition

In the context of speech recognition, HMMs are used to model the production of a sequence of sounds, accounting for acoustic, lexical, and language constraints. The discussion below focuses on the acoustic modeling aspect of HMMs.

At the lowest level, most acoustic HMMs model the production of phones. Section 2.1.1 described a simple HMM with one observation variable that could take on certain discrete values. In speech recognition, acoustic HMMs usually emit a *vector* of observations such as MFCCs, and each observation has a continuous density function described by a mixture Gaussian function. In addition, the HMMs used in acoustic modeling tend to be purely left-to-right, meaning the transition matrices are always upper triangular.

Given a set of phone-level HMMs, word-level HMMs are realized by concatenating the appropriate phone HMMs. For example, Figure 2-2 shows an HMM that could represent the word "talk," which consists of the phones *t*, *ao*, and *k*.

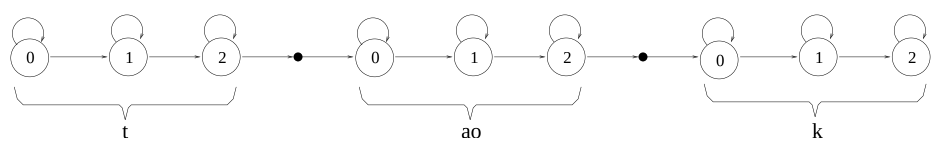

Figure 2-2: A word-level HMM for "talk"

Recognition usually consists of identifying the sequence of word models that is most likely to produce the acoustic observations of a speech waveform. In order to compare different candidate HMMs to each other, one must determine the probability of any given HMM producing a certain observation sequence. The correct way of computing this probability is to sum over all possible state sequences that could have produced the observations. However, in practice one usually compares competing HMMs using only the *best* state sequence through any given model. The Viterbi Search, described in Section 2.1.3, is an efficient algorithm for determining the optimal state sequence and its associated probability.

Note that, aside from performing recognition, one might also wish to acquire the best state sequence through an HMM for other purposes, such as performing forced transcriptions. Forced transcription is a process by which pre-determined word level transcriptions are used in conjunction with an audio waveform to produce a phone-level time alignment of the waveform. This phone-level alignment is necessary for training a recognizer.

2.1.3 Viterbi Search

Given an HMM and an observation sequence, the Viterbi search is simply an efficient way to determine an optimal state sequence through the HMM. It also yields the probability of following that state sequence and generating the given observations. More formally, given a model $\lambda = \{\pi, A, B\}$ and an observation sequence $O = \{o_0, o_1, ..., o_n\}$ then the Viterbi search can compute the state sequence $Q = \{q_0, q_1, ..., q_n\}$ that maximizes the probability $P(O, Q|\lambda)$.

The algorithm itself is based on the idea of storing only the best partial path to a state at any time. That is, for any observation o_t in O and any state i, the search remembers the best way of reaching state i at time t. The algorithm's operation is easily visualized with a *trellis* such as the one shown in Figure 2-3, which shows the best partial paths for the observation sequence $O = 0, 1, 1$. The observations are lined up horizontally, since they correspond roughly to time, and the states are aligned vertically.

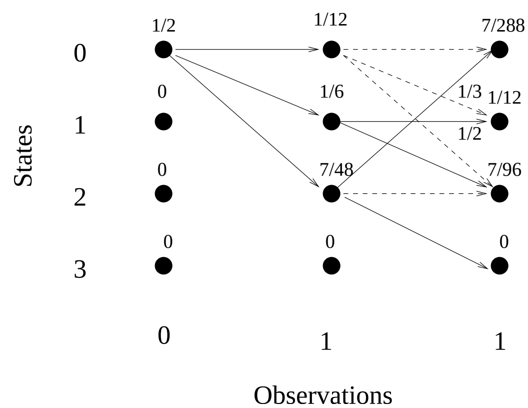

Figure 2-3: Viterbi trellis for the HMM in Figure 2.1.1

For each observation, the algorithm processes every state and determines the highest scoring partial path, based on the results from the previous observation. The maximum score of a path terminating at observation t and state i is

$$Score(t, i) = \max_{j}[Score(t-1, j) * P(q_t = i|q_{t-1} = j)] * P(o_t|q_t = i) \tag{2.2}$$

if t is greater than 0. For the states corresponding to the first observation, there is no history to take into account, so the score is simply

$$Score(0, i) = P(q_0 = i) * P(o_0|q_0 = i). \tag{2.3}$$

Computing the score in this fashion yields the values shown in Figure 2-3. For example, at the third observation the path to state 1 has two competing partial paths, one from state 0 and one from state 1. The path from state 0 would have a total score of $\frac{1}{12} * \frac{1}{3} * 1$, while the one from state 1 would have a score of $\frac{1}{6} * \frac{1}{2} * 1$. Thus, the path from state 1 is chosen. The rejected paths are shown as dotted lines in the figure, and the optimal paths are shown as solid lines.

Once the Viterbi trellis has been built, the results at the final observation are examined to find the state that has the maximum score. The optimal path can then be re-constructed by tracing backwards from that state. In Figure 2-3, the maximum score at the third observation is $\frac{1}{12}$, occurring at state 1. Tracing backwards through the trellis yields the optimal state sequence of $\{0, 1, 1\}$.

In practice, one usually uses log probability scores instead of probabilities to prevent underflow. In this case, the scores would be combined using addition instead of multiplication.

2.2 Previous Work

This section is devoted to notable audio-visual speech recognition work that has already been performed. It focuses on Multistream Hidden Markov Models and Product Hidden Markov Models, which provide a solid conceptual framework for this project. It also touches on certain ideas used in Multi-State Time Delayed Neural Networks and feature fusion via Hierarchical Linear Discriminant Analysis.

2.2.1 Multistream Hidden Markov Models

Dupont and Luettin used Multistream Hidden Markov Models as the basis for their audio-visual recognition work [2]. In this structure, a normal audio HMM such as described in Section 2.1.2 is used to model the audio stream. Concurrently, an HMM of similar structure is used to model the visual features at the same level. The two streams are processed independently by their respective HMMs, but they are forced to synchronize at pre-specified anchor points. This structure is illustrated in Figure 2-4.

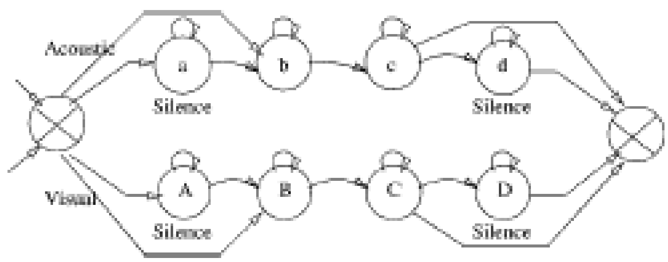

Figure 2-4: Two channel HMMs with the same topology [2]

The Multistream HMM structure is appealing because of its flexibility. It allows the two streams to be asynchronous, as long as they converge at the appropriate anchor points. The amount of asynchrony can be restricted by allowing the streams to diverge by a fixed amount. Also, each modality can be modeled with a different topology. Finally, the modeling can be performed at several different levels, such as phone, word, and utterance.

The probabilistic formulation of Multistream HMMs is based on a maximum a posteriori (MAP) strategy. The goal is to find the word string λ that, given the observations O, maximizes the following probability:

$$P(\lambda|O) = \frac{P(O|\lambda)P(\lambda)}{P(O)} = \frac{P(O^a, O^v|\lambda)P(\lambda)}{P(O^a, O^v)} \tag{2.4}$$

Note that O has been broken up into O^a and O^v, the audio and visual observations, respectively. The term P(λ) is pre-determined by the language model. The denominator can be ignored because it does not affect the choice of λ. Thus, we can focus our attention on $P(O^a, O^v|\lambda)$.

Dupont and Luettin made the common assumption that the two streams are conditionally independent, given the word string. Thus, the probability above can be broken up as follows:

$$P(O^a, O^v|\lambda) = P(O^a|\lambda)P(O^b|\lambda) \tag{2.5}$$

Although this assumption is false, it does make modeling the streams significantly easier. In order to weight the reliability of each individual channel, an exponential weight is introduced:

$$P(O^a, O^v|\lambda) \equiv P(O^a|\lambda)^w P(O^b|\lambda)^{1-w} \tag{2.6}$$

where $0 \leq w \leq 1$. Dupont and Luettin originally set w by optimizing the error rate of a held-out development set. However, in their final experiments, they derived it from the signal-to-noise ratio of the test data.

2.2.2 Product Hidden Markov Models

The product HMM is another approach proposed by Dupont and Luettin. It is simply another representation of the Multistream HMM. Here, each product state consists of several component states, one from each HMM in the multistream model. This composite model represents every possible path through the initial HMM topologies. For example, Figure 2-5 shows the product HMM corresponding to Figure 2-4. Note that the product HMM states along the diagonal represent a synchronized progression, while the off-diagonal states represent asynchrony between the two streams.

Dupont and Luettin proposed two methods of learning the asynchrony between the audio and video streams. The first method is known as static pruning. It assigns a prior probability to each of the composite states, indicating how frequently it has been visited. Then, the asynchronous states with a probability falling below a certain threshold are trimmed away.

The second method involves modeling the duration of the states in the product

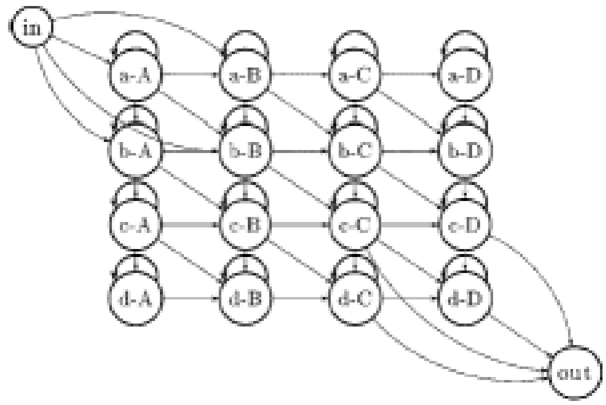

Figure 2-5: A Product HMM based on the models in Figure 2-4 [2]

HMM. Each composite state is replaced with an HMM with the topology shown in Figure 2-6. Although the states in this HMM are all identical to each other, they have different transition probabilities. Also, the loop in the last state causes the probability to decay exponentially after a certain point. In this way, we can control the expected amount of time spent in a given asynchronous state.

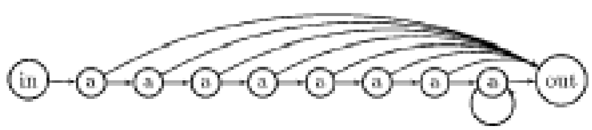

Figure 2-6: Duration model [2]

2.2.3 Multi-State Time Delay Neural Networks

Meier et al investigated the use of multi-state time delay neural networks (MS-TDNNs) in audio-visual speech recognition [6]. The MS-TDNN has a multi-layered architecture. The lowest layer is the input layer. At each frame t, the feature vectors from the frames $\{t-2, t-1, t\}$ are passed to the hidden layer. Similarly, the phoneme layer generates outputs based on five sequential frames from the first hidden layer. The use of different-sized windows in the first two stages captures behavior that takes place over varying amounts of time.

The integration experiments utilized two MS-TDNNs, one for audio and one for video. The combination occurred at the outputs of the first, second, and fifth stages. This layered architecture is quite flexible, and allows integration at several different points.

Three different techniques were used to weight the information coming from each channel. The first method used the entropy in the given modalities. The second used a piecewise-linear mapping of the signal-to-noise (SNR) ratio in the acoustic signal. The SNR estimate was adjusted every 500 ms. The last weighting scheme was the most intriguing. It utilized a neural network to estimate separate weights for each phone or viseme, based on the training data. This method makes intuitive sense, because some units will probably be hard to distinguish in the audio channel but not in the video, and vice versa.

One advantage of using the MS-TDNN approach is that it does not assume that the two streams are conditionally independent. However, choosing this approach makes it difficult to model the dependency and asynchrony between the two modalities.

2.2.4 HiLDA Feature Fusion

Most experiments conducted in audio-visual integration seem to favor late integration over early integration. Perhaps this is because late integration allows the different modalities to be weighted more effectively and permits explicit asynchrony modeling. One exception to this pattern is the Hierarchical Linear Discriminant Analysis (HiLDA) integration strategy. HiLDA is an early integration method that performs almost as well as the HMM techniques described above, and even out-performs them in a clean speech environment [8]. The HiLDA technique is illustrated in Figure 2-7.

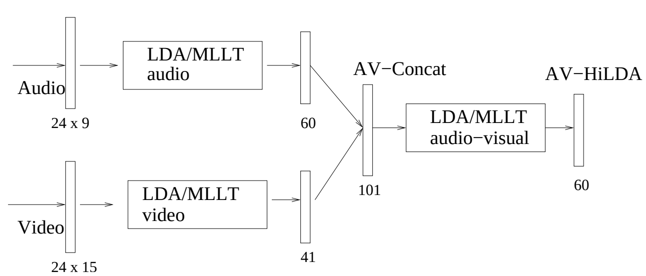

Figure 2-7: Architecture of HiLDA System

Linear Discriminant Analysis (LDA) is applied to the individual audio and visual vectors, and the resulting vectors are concatenated. Since this combined vector is quite large, it is projected to a lower dimensional space by applying LDA one more time. The result is then rotated with a Maximum Likelihood Linear Transform (MLLT). Unlike the HMM-based fusion methods described above, this technique doesn't rely on the assumption of conditional independence between the two modalities.

Chapter 3

Integration

This section describes the audio-visual integration strategy taken in the SUMMIT recognizer. First, the rationale behind choosing a late integration strategy are discussed. After that, the use of a segment network in the SUMMIT speech recognizer and its role in producing scores are introduced. Finally, the technique for resolving asynchrony between the audio and visual streams is described.

3.1 Early vs. Late Integration

Figure 3-1 gives a high-level overview of the current integration strategy, and shows how the audio and visual streams are handled. Each input stream is processed by a classifier. The output of classification is a set of phonetic units with associated likelihoods, and a set of possible viseme units. The viseme units are then mapped to phonetic units using a probabilistic mechanism such as a histogram. These two streams of phonetic units are then combined in the integration module. This process can be viewed as late integration at the phonetic level.

We chose to perform late integration for several reasons. Early integration is not very well suited to this task. Feature concatenation would result in a high-dimensional data space, making a large multimodal database necessary for robust statistical model training. Also, it would be very difficult to model the asynchrony of the two input modalities, since the system would be making decisions on the

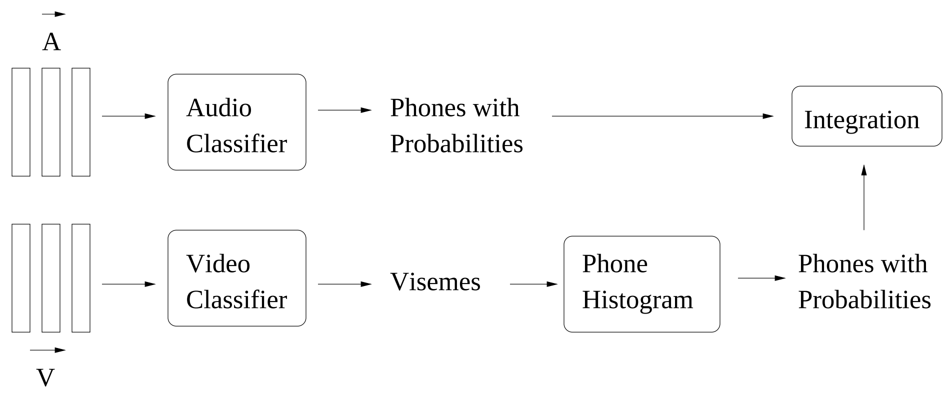

Figure 3-1: Overview of integration strategy

features themselves, not on symbols derived from those features. It may be possible to use time windows consisting of multiple frames to try and capture associated events occurring at different times, but this approach would also be difficult. In addition, explicit channel weighting would be impossible. Instead, the classifier would have to be trained on large amounts of data in various noise environments.

In contrast, late integration offers many advantages. We could make full use of the data we possess. All audio-visual speech could be used to train both classifiers, while the purely audio data we already have could be used to strengthen the audio classifier. Late integration also permits explicit modeling of the asynchrony between the two streams.

3.2 Channel Combination

The SUMMIT speech recognizer uses a Viterbi search to generate hypotheses. Two types of acoustic scores are used during the search: segment scores and boundary scores. The audio-visual integration strategy aims to augment the audio segment scores with additional scores based on segment information. The relative weighting of the various classifier scores is determined by optimization on a held out data set.

In order to understand the scoring mechanisms, one must understand the creation and use of segment networks in SUMMIT. The first subsection below provides a high level overview of segment networks. Then the subsequent subsections describe their role in producing audio and visual scores, and how these two scores are combined.

3.2.1 Segment Networks

One of SUMMIT's distinguishing characteristics is its use of segment networks to process speech. Typical speech recognizers use measurements extracted from frames that are equidistant in time. In contrast, segment networks are based on the idea that speech waveforms can be broken up into variable length segments that each correspond to a significant acoustic unit, such as a phone.

When SUMMIT begins processing a speech waveform, it hypothesizes points in time where salient acoustic landmarks might exist. These hypothesized landmarks are used to generate a network of possible segments. Figure 3-2 shows a simple segment network constructed for the phrase "computers that talk."

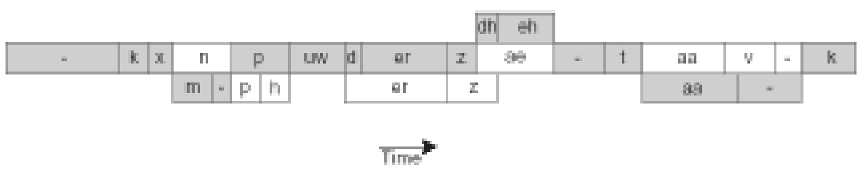

Figure 3-2: Example segment network

The end result of recognition is a path through the segment network, where all selected segments are contiguous in time and are assigned an appropriate phone. Various factors are considered when deciding on this path, such as acoustic, lexical, and language constraints. The optimal path through the network in Figure 3-2 has been highlighted.

3.2.2 Segment Scores vs. Boundary Scores

When determining acoustic constraints on its path selection, the recognizer combines scores extracted in two conceptually different ways. The first type of score involves features taken at hypothesized segment boundaries. These scores are known as *boundary scores*. The second type, which is based on measurements taken over an entire segment, are called *segment scores*. These two score categories are combined

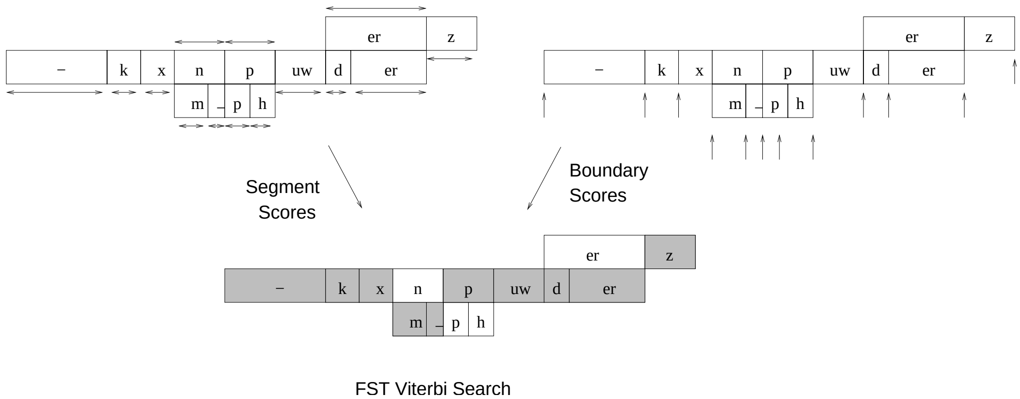

Figure 3-3: Combination of Segment Scores and Boundary Scores

in a weighted linear fashion during the recognizer's main search. This concept is illustrated in Figure 3-3.

At a high level, the audio-visual recognizer simply modifies the segment scoring scheme. The segment acoustic scores are linearly combined with segment level visual scores to produce an audio-visual score for every segment. The visual scores are generated using visual models that each span one frame of video, in contrast to the audio models that span an entire phonetic segment. In order to generate segment-level scores from frame-level models, a method for combining the frame scores must be devised. This combination method is described in the next section, as part of the modeling of the audio and visual streams' asynchrony.

3.3 Modeling Asynchrony

As discussed in Section 1.2.3, one of the key issues in audio-visual integration is the modeling of the asynchrony between the audio and video streams. In SUMMIT, this modeling consists of three main parts. First, a way of assigning fixed-length video frames to variable-length audio segments is defined, in order to meaningfully compare model scores for the two streams. The mapping is done in such a way that any segment path will include each video frame exactly once. Next, the degree of audio-visual asynchrony in a given segment is represented by a segment-level Hidden Markov Model (HMM). Finally, the most likely alignment of the streams in each segment is found by doing a Viterbi search on the HMM. The score generated for this segment is then simply the total log probability of the best path found.

3.3.1 Mapping Frames to Segments

For any given audio segment, the recognizer must identify the visual frames that correspond to it. Since the audio sampling rate is often not an integer multiple of the video sampling rate, some convention must be defined to systematically map frames to segments. Figure 3-4 shows one such convention.

In this scheme, the beginning and ending times of any frame are averaged to obtain the frame's midpoint. Then, the frame is assigned to any segment whose beginning lies before the midpoint, and whose end lies after it. Figure 3-4 shows in gray which frames have been mapped to the segment labeled *d(uw,er)*. This scheme has the desirable property that for any given path of non-overlapping segments covering all times in the utterance, each video frame is mapped to exactly one segment.

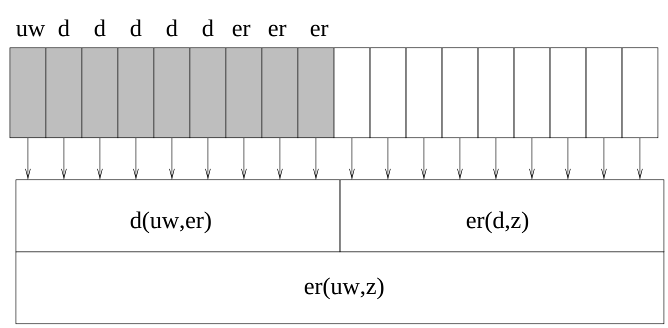

Figure 3-4: Mapping frames to segments by midpoint of frame

Note that this figure presents a different labeling scheme than that used in Figure 3-2. Here, each segment label represents a *triphone* of the form B(A,C), where B is the current phone, A is the preceding phone, and C is the following phone. This new scheme introduces contextual dependencies between adjacent phones, and will be important in the asynchrony representation discussed next.

3.3.2 Representing Asynchrony with a Hidden Markov Model

Due to the different sampling rates of the two streams, and effects such as anticipatory visual cues, it is very likely that not all the visual frames in an audio segment will represent exactly one phone. For example, in Figure 3-4, the first frame in the segment *d(uw,er)* actually represents the transition from the phone *uw*, most of the frames represent the central phone *d*, and a few of the end phones correspond to preparation for the upcoming phone *er*. In order to figure out the best possible frame alignment, we must use a probabilistic representation that will allow us to efficiently compare competing alignments. The representation used in this project is a Hidden Markov Model (HMM).

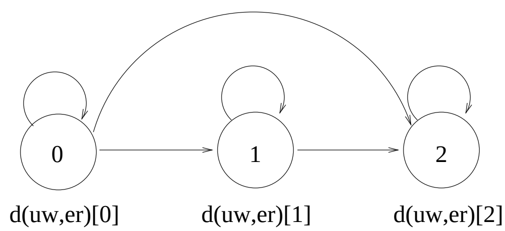

Figure 3-5: Using a Hidden Markov Model to represent audio-visual asynchrony for a given segment

Figure 3-5 shows how the asynchrony is represented for the triphone *d(uw,er)*. The HMM has three states, corresponding roughly to the transition from the left context *uw*, the main phone *d*, and the transition to the right context *er*. This figure defines yet another labeling convention, which is built on top of the triphone labels. Each triphone now has three state labels, identified by bracketed indices. In the example shown, the state labels are *d(uw,er)[0]*, *d(uw,er)[1]*, and *d(uw,er)[2]*. The structure of the HMM is left-to-right, with possible self transitions and skip states.

The initial state probabilities and transition probabilities for each HMM are trained from frame-level transcriptions of recorded data. The observation probabilities associated with each state come from frame-level visual models, which yield likelihoods for observing a certain set of visual feature values given a certain state.

Note that there is a large degree of state sharing possible in such a structure. There are roughly 234,000 triphones in a standard SUMMIT system, which yields a possible 702,000 frame-level models. However, many of these models actually represent very similar phones. For example, *d(uw,er)[0]* and *d(uw,iy)[0]* both describe a *d* transitioning from a *uw* so it might be reasonable for them to use the same model. Similar sharing can be induced for the last state in an HMM, i.e. state 2.

Aside from this index based classing, one can take advantage of the sharing across phones typically done in a recognizer, where acoustically similar diphone transitions and triphone segment models are grouped together. In this case, the diphone classing would help us share the frame models with indices 0 and 2, and the triphone classing would help us share frame models with index 1.

3.3.3 Finding an Optimal Alignment

Having established a model structure for aligning visual frames with an audio segment, it is quite straightforward to determine the optimal alignment. We simply do a Viterbi search over the HMM, and find the state sequence that is most likely to produce the visual features observed. The optimal state sequence will tell us the frame alignment preferred, and the best path score (usually computed as the total log probability of the transitions and observations along the path) will yield an overall score for the entire segment.

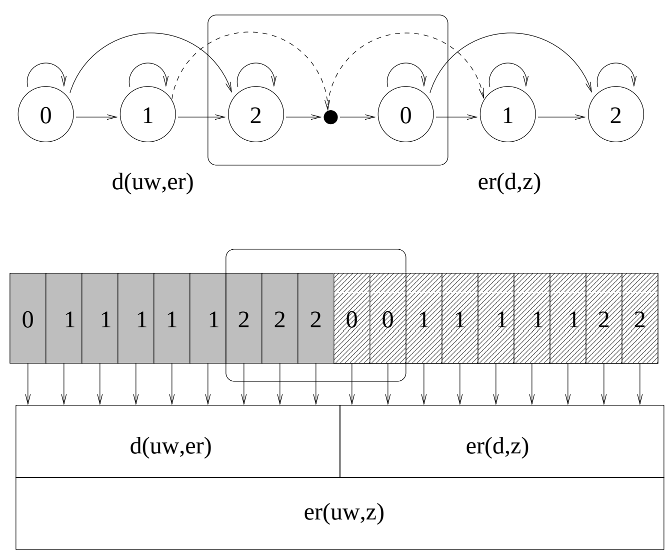

Figure 3-6: Two consecutive segment-level HMMs. The boxed HMM states and their assigned frames share the same observation model.

To illustrate how the asynchrony modeling is used in an actual segment path with multiple segments, consider the example shown in Figure 3-6. Here, the triphones *d(uw,er)* and *er(d,z)* occur one after the other. Each triphone segment is modeled with its own HMM, which has been searched to produce the alignments shown in the diagram. Note that the two HMMs have been joined with a non-emitting node, which allows transitions from state 1 of *d(uw,er)* and to state 1 of *er(d,z)*. Although the HMM structure implemented allows skip states, it may be advisable to anchor every phone at state 1 when performing actual audio-visual integration. This way, each state sequence would be forced to contain the main acoustic content of the phone, and the other two states would simply serve to resolve asynchrony with adjacent phones.

For example, consider the boundary between the two phones *d(uw,er)* and *er(d,z)*. State *d(uw,er)[2]* and state *er(d,z)[0]*, which are boxed together in Figure 3-6, both represent the transition from one phone to the next. Thus, it is possible for these states to share the same visual observation density function. The only real difference between the states are their associated transition probabilities, which depend on the HMM to which each state belongs. Together, these two states explicitly determine the amount of asynchrony at the phone boundary. By varying the spread of the frames corresponding to these states, the amount of asynchrony can be shifted and/or expanded.

Chapter 4

Experimental Study

Ideally, the end result of integration would be a full audio-visual recognizer that could be compared against the standalone audio recognizer. However, in reality the integration work described above is part of a multi-person project. At the time of this writing, the visual classifier was still unfinished. Thus, an alternative method of evaluating the integration infrastructure had to be found.

An easy way to test the infrastructure is to use audio frames in place of video frames. In this scenario, the frame-based scores were generated using the same audio waveform processed by SUMMIT's boundary classifier, and the normal audio segment scores were not used. This process was consistent with the goals of the evaluation, which were to verify the functionality of the code and to understand how well the new frame-based scoring mechanism performed during recognition. The next two sections describe the conditions under which the evaluation was performed and the results obtained from the experiments.

4.1 Evaluation Conditions

This section describes the recognizer configurations that were used during the experiments. It also details the procedure for training frame-level acoustic models and per-segment HMMs, and discusses some optimizations that were performed.

4.1.1 Recognizers

The baseline SUMMIT recognizer was configured as described in [10], incorporating both boundary and segment acoustic scores. The boundary models included 134 class triphone internal models and 643 class diphone transition models. For segment scoring, 960 class triphones were used. The classes were determined automatically using bottom-up clustering.

The test recognizer was also triphone based, but utilized boundary and frame-level scores. Recall that the frame-level models are based on a convention that maps each triphone label to three frame model labels, identified by a bracketed index of 0, 1, or 2. The frame models corresponding to states 0 and 2 were classed according to the baseline recognizer's diphone transition classing. That is, the models with index 0 were grouped by the transition from their left context to the central phone, and the models with index 2 were grouped by their central phone and right context. On the other hand, the frame models with index 1 were classed according to the baseline recognizer's triphone segment classing. In total there were 573 frame models of index 0, 960 frame models of index 1, and 573 frame models of index 2.

The frame classification was based on 14 MFCC features extracted across seven regions for each frame: three regions preceding the frame, the frame itself, and three regions following the frame. Each region was 5 milliseconds in length. Once the feature vectors were extracted, principal components analysis (PCA) was applied to decorrelate the measurements and reduce dimensionality to 50 before performing frame classification.

4.1.2 Training Frame-Level Acoustic Models and HMMs

In general, when one trains a set of acoustic models for speech recognition, a set of waveforms and corresponding time-stamped transcriptions are required. Sometimes these time-stamped transcriptions are readily available. Usually, though, only the word level transcriptions are available, and the time stamps must be generated by the recognizer via forced transcription. This situation is troublesome, because a good recognizer is needed to produce reliable forced transcriptions, but forced transcriptions are required to train a good recognizer.

Since recognition in the TIMIT domain is at the phone level, manually produced phone timestamps are available for each utterance transcription. Thus, a set of time-stamped transcriptions are already provided for training segment-level boundary models. However, the segment scores have to be produced with the aid of frame-level models, which require frame-level transcriptions. To generate an initial set of frame-level transcriptions, a simple algorithm that divides each phone-level time interval into thirds and assigns each third to a frame-level model was applied. For example, consider the phone-level transcription for "talk" shown in Table 4.1.2. Given this transcription, the algorithm produces the frame-level timestamps in Table 4.1.2, assuming a frame rate of 100 Hz or 10 ms per frame. Note that while this example uses 10 ms frames for simplicity, SUMMIT typically uses frames of 5 ms each.

Start Time (ms)	End Time (ms)	Phone
0	30	t(-,ao)
30	100	ao(t,k)
100	120	k(ao,-)

Table 4.1: Example phone-level transcription

Start Time (ms)	End Time (ms)	Phone
0	10	t(-,ao)[0]
10	20	t(-,ao)[1]
20	30	t(-,ao)[2]
30	40	ao(t,k)[0]
40	50	ao(t,k)[0]
50	60	ao(t,k)[0]
60	70	ao(t,k)[1]
70	80	ao(t,k)[1]
80	90	ao(t,k)[2]
90	100	ao(t,k)[2]
100	110	k(ao,-)[1]
110	120	k(ao,-)[2]

Table 4.2: Example phone-level transcription

The triphone *t(-,ao)* is easy to divide into thirds, since the time interval it occurs over corresponds to exactly three frames. The *ao(t,k)* segment, on the other hand, cannot be divided evenly, so an extra frame with index 0 is inserted. The final triphone *k(ao,-)* represents a special case, where the total number of frames across the phone is less than three. In this case, we start building up from index 1, to ensure that the main phone is included. Thus a one-frame triphone A would have frame transcription A[1], and a two-frame version of A would have transcription A[1] A[2], as shown.

In this simple example, the phone boundaries in the original transcription were all a multiple of the frame size (i.e., 10 ms). However, since the time stamps are manually determined, there are actually many boundaries that do not line up with the frame divisions. To handle this situation, we simply follow the mapping convention described in Section 3.3.1 and assign frames to phones based on their midpoint. All experiments described in this chapter used the initial frame-level transcriptions for training. That is, no iterative retraining based on forced transcription was performed.

As for the per-segment HMMs, each model's initial state probabilities and transition probabilities were determined by collecting statistics from the frame-level transcription files. When no statistics could be gathered for a particular HMM, it was initialized with a simple uniform initial state matrix and a uniform left-to-right transition matrix, as shown in Equation 4.1. The per-state observation probabilities are generated by the frame-level models, so there is no need to explicitly define them.

$$\pi = \begin{bmatrix} 1/3 & 1/3 & 1/3 \end{bmatrix} \quad A = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 0 & 1/2 & 1/2 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

4.1.3 Optimizations

When the integration strategy described in Chapter 3 was initially tested, the recognizer consumed an enormous amount of memory and ran exceedingly slowly. It required over 1 GB of RAM and took about 20 minutes to decode a single utterance. In order to reduce these computational demands, several optimizations were performed on the system. Although some of them resulted in slightly sub-optimal performance, they allowed us to perform a more thorough evaluation by running more experiments.

One way to speed up the decoding is to tighten the pruning of the recognizer's search. Originally, the recognizer used a relative pruning threshold of 30, and kept a maximum of 10,000 competing hypotheses at any point in time. These thresholds were reduced to 10 and 500, respectively. Since tightening these constraints resulted in very little performance degradation in the baseline recognizer (approximately .2%), it seemed reasonable to apply the reduced thresholds across all experiments.

In addition, since the optimal frame-level alignments for each segment are only necessary when performing forced transcription, the code was modified to only record the best path during forced transcription. This simple modification not only resulted in improved speed, but saved a large amount of memory.

Another method for reducing the requirements for both memory and speed is to simply restrict the triphone models that the search can score against. Of all 234,000 possible triphone models, only about 75% are physically realizable. Thus, prohibiting the search from considering any of the remaining 25% was a straightforward way to improve the decoding efficiency.

Implementing the optimizations described above reduced the memory requirement to about 700 MB and resulted in a decoding time of 4 minutes per utterance. Although these figures are still much worse than those of the baseline recognizer, to some degree an increase in computation is inevitable because of the scoring of multiple frames in each segment, and the large number of possible triphone models for each frame considered.

4.2 Testing and Results

The recognizer was evaluated on the TIMIT corpus [11], which defines the phonetic recognition task described in [4]. The TIMIT training set of 4620 utterances was used for training the models, and the development set of 400 utterances was used for testing. Several experiments were conducted, in which only the segment-based boundary scores were used, only the frame-based scores were used, and both scores were used in conjunction. In addition, a series of experiments were performed using full-forward scoring over the per-segment HMM instead of Viterbi scoring. The results of these experiments are shown in Table 4.2, which displays the recognizer's word error rate (WER) under various testing conditions. The performance of the baseline recognizer, which uses standard segment and boundary scores, is also shown. Note that the search methods "Viterbi" and "Full-Forward" only apply to frame scoring.

Scoring Method	Viterbi WER	Full-Forward WER
Boundaries	29.4	29.4
Frames	30.0	31.6
Boundaries + Frames	26.7	26.7
Boundaries + Segments	27.7	27.7

Table 4.3: Performance of different scoring techniques

In isolation, the frame-based scores don't perform as well as the boundary-based scores. However, when boundary scores are combined with frame scores, the resulting word error rate is better than that of either technique in isolation. In fact, it even improves on the current baseline recognizer, which uses boundary and segment scores. Finally, the full-forward scoring required slightly more computation than the Viterbi scoring, but yielded slightly worse results.

Chapter 5

Audio-Visual Data Collection

The preceding chapters have described work that has been done to prepare for audio-visual integration. Aside from having the necessary mechanism for combining the audio and visual streams, integration would also require a large amount of data for training and testing. Unfortunately, there were no publicly available databases of audio-visual speech that were suitable for our task. Therefore, we initiated a fairly large data collection effort in anticipation of the eventual integration. The following sections describe the main data sets available, our data collection procedure, and some statistics on the final set.

5.1 Existing Data Sets

The most extensive database collected specifically for audio-visual speech recognition (AVSR) research is the IBM ViaVoice database [5]. It contains 50 hours of dictation style utterances, collected across 290 subjects. Unfortunately, this data set has not been made available for public use, and therefore could not be leveraged for our project.

Another well known data set is the Multi-Modal Verification for Teleservices and Security applications (M2VTS) corpus [7]. It was collected at the University of Surrey in England, and included 4 recordings of 295 subjects. Since this database was geared towards security applications, the data consisted mainly of continuous digit sequences read into a camera. It also included rotating head shots, i.e. video taken from different angles, which are appropriate for 3D imaging experiments. This database is available at cost from the University of Surrey, but the use of digit sequences was too constraining for our purposes. The newest version of the database included some non-digit utterances, but not enough to robustly train and test a recognizer. Several smaller databases encountered online had similar limitations, and thus are not described here.

5.2 AVSR-TIMIT Data Collection

The database for this project consisted of read utterances from the TIMIT-SX data set [11]. Volunteers were taken into a quiet room and asked to record a set of sentences, which were all recorded and digitally stored.

The first utterance recorded by any volunteer was "Don't ask me to carry an oily rag like that." This sentence was simply provided to make the user more accustomed to the recording procedure, and will be ignored when generating the final corpus.

5.2.1 Goals

The main goal of the data collection was to collect a large degree of phonetically varied speech, balanced over the entire corpus. The SX sentences within the TIMIT corpus are a good match for such a requirement. TIMIT SX contains roughly 450 sentences designed specifically to cover the most common phonetic contexts in as few sentences as possible.

In addition to phonetic variety, another goal was to include a large number of speakers. Ideally, the speakers would include a roughly even split between males and females, and be fairly fluent in English. The recording conditions were designed to be visually and acoustically clean. However, the use of a far-field microphone instead of a close-talking one introduced some degree of environmental noise and room reverberation.

5.2.2 Recording Process

The volunteer was seated in front of a computer, which had a beam forming microphone array on the keyboard and a video camera mounted on a tripod, behind and slightly above the monitor. The volunteer was instructed to interact with the computer using a graphical interface, shown in Figure 5-1.

Figure 5-1: Graphical Interface for Data Collection

Each of the roughly 20 utterances was shown on the interface in turn. The user would press and hold the "Record" button to record the utterance. When the button was released, the program would echo the recorded waveform back, so that the user could hear his/her own recording. After that, the program would automatically advance to the next utterance.

Throughout the entire process, someone was present in the room to tell the user to re-record a sentence if necessary, e.g. if there was an orthographic mistake. If a sentence had to be re-recorded, the user would simply use the "Backup" button shown in Figure 5-1 to revert to the previous utterance. This practice was necessary to ensure that the speech matched the orthographic transcription so that retranscription would not be necessary.

The video camera had a side panel that would allow the user to see if his/her face was still in frame. Since the camera was mounted slightly above the monitor and the subject was reading from the screen, his/her eyes were usually directed slightly downwards. A blue curtain was draped behind the user to make the background mostly uniform.

(a) Without side illumination (b) With side illumination

Figure 5-2: Sample screen shots from audio-visual data collection

For the last five utterances in each session, a small lamp was turned on so that the subject's face was illuminated from the side. This practice was included to add some lighting variation to the data. Figure 5-2 shows two sample screen shots from the audio-visual data set collected.

5.2.3 Statistics

The final audio-visual TIMIT data set included 223 speakers, of which 117 were male and 106 were female. All but 12 of the subjects were fluent speakers of English. Since each person recorded about 20 sentences, the set consists of roughly 4500 utterances. The video was recorded in uncompressed DV (digital video) AVI format in 720 x 480 resolution.

Chapter 6

Conclusion

6.1 Summary

The goal of this thesis was to develop a speech recognition system that could use visual as well as audio data, in an effort to improve recognition performance. The improvement would be mild in clean acoustic environments, and more substantial in noisy environments.

The infrastructure necessary to integrate the audio and visual channels was developed. Video frames were mapped to audio segments, and the asynchrony between the two streams was resolved using a Hidden Markov Model for each segment. A video score was then generated using a segment-constrained Viterbi search, and was combined with the audio score to produce an audio-visual score used by the recognizer's main search.

In the absence of a video classifier, the infrastructure was tested using audio data for both channels. The use of segment-based boundary scores and frame-based scores yielded better recognition results than the baseline recognizer on a TIMIT-SX development set of 400 utterances. Unfortunately, the large amount of computation required to use the frame-based scores make them inappropriate for real-time speech recognition tasks. However, the frame-based scoring mechanism does seem to be a good foundation for eventual integration with a visual channel.

6.2 Future Work

Although a preliminary evaluation of the frame-level scoring mechanism was performed in this thesis, there are many more experiments that would provide more insight into its performance. For example, one could finely adjust recognizer parameters such as the segment transition weight and the relative weighting between segment and boundary scores to further improve the recognition performance. Another possibility is to modify the Viterbi search over each segment's HMM to proceed in a greedy fashion, so that it only extends the best partial path at any given time. This approach could potentially increase speed while not drastically affecting the word error rate. In addition, one could try iterative retraining by generating frame-level forced transcriptions and training new models on those transcriptions.

Of course, the most promising direction for further research is using the infrastructure to integrate SUMMIT with a visual channel. This direction is promising from several perspectives. As mentioned in Chapter 1, using visually derived frame-based scores would provide new information that could noticeably improve recognition performance, especially in noisy environments. In addition, the speed would be greatly improved because the video sampling rate is usually around 30 frames/sec, compared with the current audio sampling rate of 200 frames/sec. This slower sampling rate would improve decoding speed by roughly an order of magnitude, since there will be less frames to score in each segment. Also, since audio-visual asynchrony usually comes in the form of anticipatory visual cues, the per-segment HMM could be reduced to two states, so that only the current phone and the right context need be considered. Furthermore, since there are less visemes than phones, there would be fewer unique models to score against. These factors, combined with further optimization, could result in an audio-visual recognizer that yields better recognition performance than the current baseline and decodes in real time.

Appendix A

Phonetic Unit Chart

The following is a table of the IPA symbols for the phones and the corresponding TIMIT labels, along with example occurrences. The table continues on the following page.

IPA	TIMIT	Example	IPA	TIMIT	Example
[ɑ]	aa	b*o*b	[ɨ]	ix	deb*i*t
[æ]	ae	b*a*t	[i]	iy	b*ee*t
[ʌ]	ah	b*u*t	[ǰ]	jh	*j*oke
[ɔ]	ao	b*ou*ght	[k]	k	*k*ey
[ɑ^{w}]	aw	b*ou*t	[k$^{□}$]	kcl	k closure
[ə]	ax	*a*bout	[l]	l	*l*ay
[ə^{h}]	ax-h	p*o*tato	[m]	m	*m*o*m*
[ɚ]	axr	butt*er*	[n]	n	*n*oo*n*
[ɑ^{y}]	ay	b*i*te	[ŋ]	ng	si*ng*
[b]	b	*b*ee	[ɾ̃]	nx	wi*nn*er
[b$^{□}$]	bcl	b closure	[o]	ow	b*oa*t
[č]	ch	*ch*oke	[ɔ^{y}]	oy	b*oy*
[d]	d	*d*ay	[p]	p	*p*ea
[d$^{□}$]	dcl	d closure	[□]	pau	pause
[ð]	dh	*th*en	[p$^{□}$]	pcl	p closure

[ɾ]	dx	mu*dd*y	[ʔ]	q	glottal stop
[ɛ]	eh	b*e*t	[r]	r	*r*ay
[l̩]	el	bott*le*	[s]	s	*s*ea
[m̩]	em	bott*om*	[š]	sh	*sh*e
[n̩]	en	butt*on*	[t]	t	*t*ea
[ŋ̩]	eng	Wash*ing*ton	[t□]	tcl	t closure
[□]	epi	epenthetic silence	[θ]	th	*th*in
[ɝ]	er	b*ir*d	[ʊ]	uh	b*oo*k
[e]	ey	b*ai*t	[u]	uw	b*oo*t
[f]	f	*f*in	[ü]	ux	t*oo*t
[g]	g	*g*ay	[v]	v	*v*an
[g□]	gcl	g closure	[w]	w	*w*ay
[h]	hh	*h*ay	[y]	y	*y*acht
[ɦ]	hv	a*h*ead	[z]	z	*z*one
[ɪ]	ih	b*i*t	[ž]	zh	a*z*ure
-	h#	utterance initial and final silence			

Bibliography

[1] C. Benoit. The intrinsic bimodality of speech communication and the synthesis of talking faces. In *Journal on Communications of the Scientific Society for Telecommunications*, Hungary, number 43, pages 32–40, September 1992.

[2] S. Dupont and J. Luettin. Audio-visual speech modeling for continuous speech recognition. In *IEEE Transactions on Multimedia*, number 2, pages 141–151, September 2000.

[3] J. Glass. A probabilistic framework for segment-based speech recognition. To appear in *Computer Speech and Language*, 2003.

[4] A. Halberstadt and J. Glass. Heterogeneous measurements and multiple classifiers for speech recognition. In *Proceedings of ICSLP 98*, Sydney, Australia, November 1998.

[5] IBM Research - Audio Visual Speech Technologies: Data Collection. Accessed online at http://www.research.ibm.com/AVSTG/data.html, May 2003.

[6] U. Meier, R. Stiefelhagen, J. Yang, and A. Waibel. Towards unrestricted lip reading. In *International Journal of Pattern Recognition and Artificial Intelligence*, number 14, pages 571–585, August 2000.

[7] K. Messer, J. Matas, J. Kittler, and K. Jonsson. XM2VTSDB: The extended M2VTS database. In *Audio- and Video-based Biometric Person Authentication, AVBPA'99*, pages 72–77, Washington, D.C., March 1999. 16 IDIAP–RR 99-02.

[8] C. Neti, *et al.* Audio-visual speech recognition. In *Technical Report, Center for Language and Speech Processing*, Baltimore, Maryland, 2000. The Johns Hopkins University.

[9] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, number 2, pages 257–286, February 1989.

[10] N. Ström, L. Hetherington, T.J. Hazen, E. Sandness, and J. Glass. Acoustic modeling improvements in a segment-based speech recognizer. In *Proc. 1999 IEEE ASRU Workshop*, Keystone, CO, December 1999.

[11] V. Zue, S. Seneff, and J. Glass. Speech database development: TIMIT and beyond. In *Speech Communication*, number 9:4, pages 351–356, 1990.