Integrated Audio-Video Synchronization System for use in Multimedia Applications

Mohamed El-Helaly

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

November, 2006

Your file Votre référence
ISBN: 978-0-494-28913-6
Our file Notre référence
ISBN: 978-0-494-28913-6

Abstract

Integrated Audio-Video Synchronization System for use in Multimedia Applications

Mohamed El-Helaly

The use of multimedia system have moved beyond the studio barriers and into the homes. As computers become more powerful, multimedia systems become more realizable on the PC. As these multimedia systems become more complicated, the need to provide complex integration systems and synchronization arises.

To develop a multimedia system, one must ensure that a synchronization approach is in place to solve the timing issues related to the media types involved. Temporal information in multimedia systems must be maintained such that no loss of coherency is endured. The system must ensure that no matter how much processing is performed on the signals, the output has to maintain the temporal integrity of the signals as they were when they were inputted.

This thesis develops a multimedia system that processes two media streams. Audio and video streams are fed to the system. The system produces an object segmented output, (silhouettes of the object) along with the recognized speech from the audio. The speech that is to be recognized by the system is spoken by the objects/speakers. The challenge lies in maintaining the synchronization and integrating the video and the recognized speech at the output. Note that the system is a stream based system by that the video and audio are continuously captured and processed.

This thesis presents a solution to the problem of synchronization in the temporal domain and the overall integration of the multimedia system. The thesis presents a time-stamp approach to solve the synchronization problem between audio and video signals. This approach is adaptive to the cases where the video processing delay is larger or smaller than the audio processing delay. The contributions include the

verification of using time-stamps in the synchronization process and that it is possible to synchronize heavily delayed signals. The system requires an integration process such that the audio and video signals are integrated with one another at the output.

Acknowledgments

I am truly thankful to my family for their continued support towards my education. Thank you Dad, Dr. Ahmed El-Helaly, and thank you Mom, Dr. Siham Sharawy. I am forever grateful to your support and love.

My special thanks to my colleagues in the ECE department, to Rabih Mahzoub supervised by Dr. O'Shaughnessy from the INRS in developing the speech recognition system, and to the 490 group in helping gathering the results.

Finally I would like to thank the VidPro group at Concordia University for their help, and Dr. Aishy Amer for her guidance in completing this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations and Notations

Notation	Meaning
A	Speech Information (acoustic data) to be recognized
a_i	Sequence of acoustic symbols that make up A
W	String of words
$\hat{W}$	Recognized Word String
$\mathcal{V}$	Vocabulary of words
S_i	Markov Chain States
$\Phi(v)$	Phonetic pronunciation of the word v
$x[n]$	Discrete Speech Signal
fs	Sampling Frequency
τ_{frame}	Processing Time for One frame
$FrameCount$	Number of Frames Processed
τ_{frames}	Processing times which are accumulated to represent the total time of processing $FrameCount$ frames
T_{frame_j}	Time-Stamp for Frame j
F_R	Frame Rate in frames per second
$\triangle A(i)$	True time of speech up to utterance i, not including delay times
SIL	Silence time and recognizer delay
$\tau_{s(i)}$	System time-stamp of file i
$\tau_{u(i)}$	Time Of Utterance
$SyncFrame$	Number of Synchronization Frames
SF	Number of Synchronization Frames
$Ph(i)$	Phonemes of utterance i
$SF(i)$	SyncFrame i
DF	Difference Frames between the actual frame to insert and the synchronization frame
Th	Error Threshold

Abbreviation	Meaning
A/D	Analog to Digital
ASR	Automatic Speech Recognizer
DCT	Discrete Cosine Transform
DEFSM	Dynamic Extended Finite State Machines
DFT	Discrete Fourier Transform
DSs	Description schemes
DTS	Decoding Time-Stamp
DTSM	Dynamic Timed Synchronization Model
Eq	Equation
FFT	Fast Fourier Transform
FSM	Finite State Machine
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
IDFT	Inverse Discrete Fourier Transform
IPL	Image Processing Library
ITU	International Telecommunication Union
LVCSR	Large Vocabulary Continuous Speech Recognition
MDS	Multimedia Descriptor schemes
MFCC	Mel Frequency Cepstral Coefficients
mlf	Master Label File
MU	Media Unit
NIST	National Institute of Standards and Technology
PDA	Personal Digital Assistant
PTS	Presentation Time-Stamp
QoS	Quality of Service
RF	Radio Frequency
TBC	Time Base Corrector
TDOA	Time delay of Arrival
TPT	Target Play Time
Sec	Section
VIDAS	Video Assisted Audio Coding and Representation

Chapter 1

Introduction

Multimedia applications have become very popular in our world today. Applications involving audio and video as a data medium are present in industries such as television, film and even visual arts. It follows that there is always a need to improve the methods and technology that makes these applications possible.

This thesis is focused on developing the methods of one such application. The application involves synchronizing two media streams: audio and video. The system is meant to operate as an on-line system, where one would be able to observe his/her silhouette while observing the recognized speech him/her uttered. The objective is to focus on the video objects and the relative speech and relate one media stream to the other. The work is useful in synchronizing media streams in the ever growing world of connectivity and to provide solutions to interactive multimedia applications.

1.1 Objectives and Problem Statement

This thesis is focused on synchronization methods of processed audio and video streams using the timing information of each one. The timing information provides a temporal relationship between the streams as long as it is obtained accurately. The

audio and video streams are processed before the synchronization which incurs a delay which is not uniform. The processing involves detecting the objects in the video and recognizing the speech uttered by the objects.

The thesis has two main objectives:

1. To develop a real-time multimedia system using two media streams. These media streams used are audio and video streams. The streams are to be processed in parallel and integrated. This involves developing a communication system such that the streams are controlled.

2. To synchronize the streams and relate them to one another. The streams are separated at capture time and so an algorithm is proposed to achieve media synchronization. Synchronization is achieved by using the timing information from the media streams.

The first challenge of the system lies in the integration of the streams. The media streams are separated and processed before any output takes place. A system must be developed to integrate both streams to one another. The need of integration comes from the fact that the media streams are separated at capture. The streams are decoupled so that separate processing on them can take place. If the integration of the streams is not performed, the output would be incoherent. The system's media streams have to be stored and controlled such that their integration is possible. The problem of integration is closely related to the synchronization problem. If there is no need for synchronization then the integration of the streams would not be needed as well.

Another challenge of the system is the synchronization of the media streams. Synchronization between audio and video signals is an essential aspect of multimedia applications. Applications that involve the use of audio and video streams have

very strict timing requirements. Therefore, multimedia applications require a synchronization scheme to be able to transmit and receive them in a coherent manner. The synchronization scheme is responsible in ensuring that the audio and video are synchronized in the same way as they were when leaving the real-life object. The synchronization problem becomes more complicated when the media streams incur processing. The processing modules add their delay during runtime and so there must exist a proposal to synchronize these delayed signals if they are to be represented in a comprehensible form.

The challenge in developing such a system lies in the synchronization process. Both audio and video streams are independent from one another and they are one object in the real world. As soon as the processing of each medium is performed, they are separated. This causes the media to be desynchronized due to the variation in processing times.

Engineers have used video information such as lip movements to synchronize the audio associated with those lips. The use of lip movements to synchronize audio and video streams is not useful for the proposed system as one of the objectives is not to use any video object information, including lip movements for the synchronization. This is because further processing of the video slows the system down and that needs to be avoided. The proposed system in this thesis is intended to provide new methods in synchronization of audio and video. The methods involving synchronization using video information have been researched extensively in literature. One main objective of the proposed system is to develop a fast synchronization system and therefore, the decision was to develop a system that did not use video information for the synchronization process.

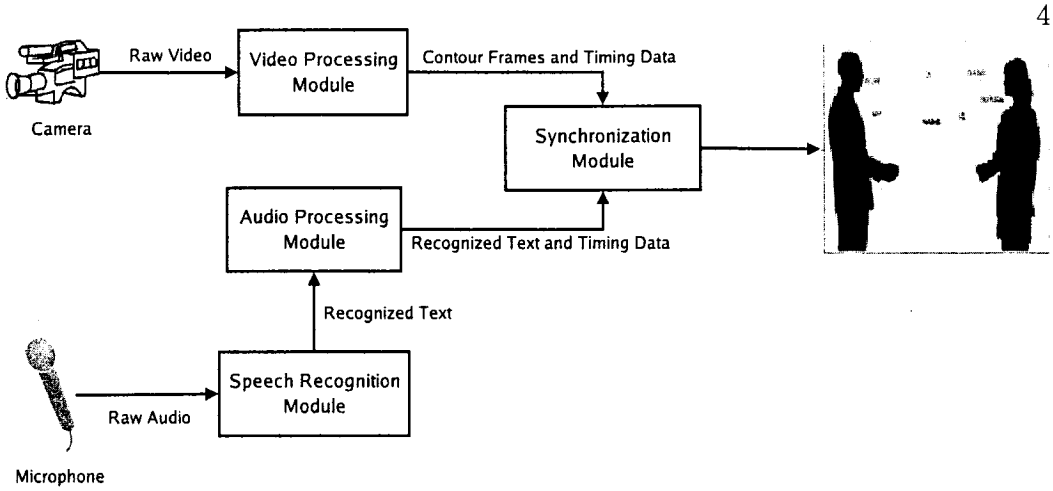


Figure 1.1: Overall System Diagram.

1.2 Overview of the Proposed System

Figure 1.1 illustrates the main components of the proposed system. The video is captured using a USB camera while the audio is captured using a microphone. The video processing modules are responsible for the object segmentation, i.e., drawing the silhouettes of the objects, while the speech recognition module is responsible for extracting the speech in the form of text. The synchronization module is responsible for integrating and synchronizing the processed video and audio streams. It is important to note that the system is a stream based system. This means that the video and audio are continuously captured and processed.

As seen in Fig. 1.1 the proposed system has two starting points in the sensors, the microphone and the video camera. The microphone's raw audio is captured by the speech recognition module and the video camera's frames are captured by the video processing module. The video processing module produces the object contour frame and the relative timing information (Sec. 4.3.1 and 5.3). The speech recognition module produces the recognized text (Ch. 3) from the raw audio frames and relays that information to the audio processing module (Sec. 5.2). The audio processing module, produces the timing information (Sec. 4.3.2) needed for synchronization. The

main module is the synchronization process and this is where the challenge and the contributions of this thesis lie. The synchronization module (Sec. 5.4) is responsible for using the timing information of the audio and video modules and their relative data to produce a synchronized sequence of contour video and recognized text, i.e., the synchronization and integration of the media streams.

The synchronization approach proposed in this thesis involves using the timing information of the audio and video streams, in the form of time-stamps. The time-stamps give the system the required temporal information needed to synchronize the signals. For the synchronization process to be successful, a system of buffers must be developed such that the audio and video streams are stored before being displayed in a synchronized fashion.

The system is integrated using a buffer system and communication is achieved between the streams using a flag system. The buffer system would store the media streams, and the flag system would determine which buffers would be flushed or not for the output of the system.

1.2.1 System Module: Speech Recognition

The speech recognition module is responsible for the audio capture from the microphone and to translate the raw audio into text. This is accomplished by the Hidden Markov Model tool available to researchers and developed by the University of Cambridge. To develop a speech recognizer from these tools, the resources must be created and recognizer must be trained. The resources include generating grammars, word networks, and dictionaries. The recognizer is then trained using these resources then adapted to the speaker's voice. This is all verified during the testing process to ensure that word recognition accuracy is up to par with the requirements of the system. The objective in this thesis is not to achieve a robust real-time recognizer, but one

which accomplishes recognition with a low processing overhead (minimum delay) and that produces comprehensible output. The recognizer produces the text from the raw audio. This is the text that is outputted onto the video sequence. The process of developing the recognizer as well as speech recognition theory is explained in Appendix A and Ch. 3 respectively.

1.2.2 System Module: Audio Processing

The audio processing module is responsible for extracting timing information from the recognizer and the subsequent output text that is useful in the synchronization process. This is accomplished by the aid of time-stamps, that provide the necessary information to determine when the audio was uttered and therefore, relate it to the video frames. The process of extracting this information is explained (amongst other concepts) in Ch. 4.

1.2.3 System Module: Video Processing

This module captures the raw video from the camera and processes each frame to produce a contour frame. Since this is real-time processing, each frame is processed as it is captured. The background image is isolated and processed independently. This is because the following frames' processing is dependent on the background frame. This means that every time the capture module captures the current frame, the frame is processed immediately, i.e., no buffering is implemented between the capture and the processing. The final output of this module is the contour image of the video. It is possible to output other versions of the video, such as the difference image or the edge image, but it is not needed as far as the graphical representation of the system is concerned. The theory behind the edge detection and the contour tracing is described below.

The timing information of the video frames (e.g., frame rate) is also extracted in this module. This timing information is to be used in the synchronization module. The implementation of this module is explained in Ch. 4.

In this section of the thesis, the background on the video processing is explained. More specifically, the object segmentation (edge detection) algorithm is analyzed and explained. Object segmentation is the process by which the objects in the frame are separated from the background. The contour image is also required for this system.

Since there are several tasks that must be accomplished in order to achieve segmentation, the algorithm used can be divided up into sub-blocks. These sub-blocks consist of motion detection, thresholding for motion detection and binary edge detection [2].

Motion Detection

The approach for motion detection is to take the difference between the current image and a pre-stored background image called a reference image. This method leads to good object detection because any object is detected regardless of whether or not it is moving. Also, uncovered areas of the background are not detected. However, this method may have a tendency to detect object shadows or object reflections that may cause enough of an illumination change on the background to be picked up [2].

Thresholding

The thresholding algorithm uses the difference image which is first divided up into K blocks, with each block having dimensions of $W \times H$ pixels. Each block is then looked at individually and a gray-level histogram is tabulated. This gray-level histogram represents the frequency of appearance of each of the possible pixel gray-level values. The histogram is then divided up into L equal intervals. In each interval, the most frequent gray level, g_{pl}, is determined. The average gray-level value, μ_k, of the whole

block is also taken. The values g_{pl} and μ_k are averaged out for all blocks and the threshold value T_g is determined [2].

Binary Edge Detection

Binary edge detection consists of performing a sequence of logical operations to determine the edges of the object. The method from [2] consists of using a 2×2 square pixel kernel to scan through the image and a 3×3 expanded kernel to determine what operation on the pixels is required. The 2×2 square pixel kernel begins at the top left corner of the image and scans its way across the rows. If there are any black pixels within the kernel, the kernel either sits on top of an area where no object is present (all 4 pixels are black) or it sits on top of an object edge (1, 2 or 3 pixels are white). In either case, none of the pixel values should be changed since the black background should remain black and any pixels that represent the edge of an object should remain white. If however, the 2×2 kernel covers an area where all 4 pixels are white (all 1's), scanning must temporarily stop and the kernel must be expanded to 3×3 to determine where the real edges lie.

1.2.4 System Module: Media Synchronization

This module involves implementing methods in synchronizing audio and video signals after they have been processed. Audio and video signals in the system are processed separately and their processing time do vary. The challenge is to synchronize the audio and video signals after processing with knowledge of the processing time of the two signals. The input to this module are the audio and video processed outputs from their respective modules as well their timing information. The obstacle as mentioned in Sec. 1.1 is that the processing modules add their delay and so there must exist an approach to synchronize these delayed signals, if they are to be outputted in a

comprehensible form.

The synchronization logic is meant to compensate for the delay in the video and audio processing modules and to realign the signals in time using their temporal information. The output is displayed and consists of the contour image and the recognized text flowing from one object to the other. This module is explained in more detail in Ch. 4.

1.3 Contributions

The main contribution of this thesis is the development of a synchronization method for a multimedia application[1]. The application itself involves three main modules: audio processing module (includes the recognizer), video processing module (to produce the silhouettes of the objects) and the synchronization module. The interaction of these modules is complex so an intricate communication system between them had to be developed which adds to the contribution of this thesis.

Note that the development of a speech recognition system is not trivial and a fast system was developed. Even though the recognizer does not operate in real-time, the significance of its development cannot be ignored.

1.4 Thesis Outline

The thesis begins with Ch. 2 which provides the literature review which describes other work in the field of media synchronization and integration. The speech recognition module is described in Ch. 3 and Appendix A. This module and appendix highlight the process of building a recognizer, from training the system to testing

[1] A paper based on the proposed system was published at the Canadian Conference on Electrical and Computer Engineering 2006 (CCECE06). Two more papers are also submitted to the IEEE journal of Transactions on Multimedia and the International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007).

it. Chapter 4 describes the synchronization approach and explains the methodology behind it, while Ch. 5 explains the integration of the system. The results of the system which explain the limits of the system and the fact that synchronization was successful are displayed in Ch. 6. Finally the conclusion is presented in Ch. 7 and it summarizes the main contributions and the future work that could be added to the system.

Chapter 2

Review: Audio-Video Synchronization

This chapter provides a summary of the research that already exists in the field of audio - video synchronization and integration.

Synchronization between audio and video is needed in multimedia applications which involve processing audio and video signals separately. An example is a audio and video recording studio where the audio and video tracks are recorded separately and therefore, a synchronization system is needed.

Synchronization research and development is being carried out on both the industrial and academic levels. Industry research is focused on maintaining synchronization within acceptable boundaries. The research provides the limits to the acceptable errors allowed in synchronization. Academic research is focused on developing methods for synchronization, without employing limits on timing errors. Academic research stated that the synchronization error limits were application dependent and not standardized as opposed to the industry standards.

2.1 Industrial Contributions

At Front Port Digital Jay Yogeshwar[3] discusses the various causes of loss of synchronization and prevention methods. Yogeshwar[3] claims that the first cause of timing error is at the video capture level. In order to synchronize multiple video sources, hardware solutions such as external sync, generator locks and frame-sync are used to integrate the synchronization pulses contained in the video signals. Time Base Correctors (TBCs) are used to synchronize the tape machine with the other signals in the studio by creating a stable video timing signal[3]. All video sources are synchronized to a master sync generator. This method is applied to tape recordings. In [3] hardware is used to align the signals according to the master sync generator.

Yogeshwar[3] explains that a second cause of timing error occurs at the processing and packeting level. Delays in the video can be due to the need to store sufficient video information in memory prior to filtering (noise reduction or otherwise). Delay compensation of the audio can be built into the TBCs, the frame synchronizers, and the noise reduction filters [3]. Multiplexing or packeting can also introduce timing errors.

The conclusion in [3] is that once the audio and video are divided into time-stamped packets, synchronizing of the two is simple provided the data is recorded to some storage device. Difficulty arises in the cases where transmission systems that are prone to delays are used. The last cause of timing error is at the presentation level or the output of the signals. In [3], it is also concluded that software playback is not as precise as hardware playback in terms of lip synchronization. This is due to the fact that software players depend on two hardware elements, the sound and VGA processors. Faster processors and tightly integrated graphic and audio processing within a workstation are key to resolving the issues of software playback precision.

Linear Acoustic Inc.[4] presents some standards of audio-video synchronization

in industry. The International Telecommunications Union (ITU) recommends that the tolerance from the point of capture to the listener should be no more that 90 milliseconds leading video to 185 milliseconds audio lagging behind video. While the video frames are being processed, it causes a delay with respect to the audio processing. This delay is recommended to be compensated into the audio path. Another recommendation from the ITU is that processing time should be indicated on both audio and video signals (in milliseconds) with its range indicated. In the case of a variable delay, there should be a signal that controls the audio delay according to the processing time calculated. Linear Acoustic Inc.[4] also explain audio-video synchronization in MPEG-2. Each audio and video signal has a Presentation Time Stamp (PTS) that enables the decoder to reconstruct the synchronized sound and video frames. After the decoder receives the audio and video data ahead of the PTS values, it properly uses these values to correctly present audio and video streams.

It is recommended in [4] to use the timestamps to synchronize audio and video signals. However, the paper lacks synchronization schemes of processed signals and only recommends that the signals be synchronized before passing it to the next stage.

The white paper prepared by Stradis Inc.[5] is another example of the industrial solutions to the synchronization problem. They present a synchronization windows application that integrates MPEG-2 decoders into the video system. They employ a buffer system to store audio and video packets and frames. They use timestamps to correctly align the packets and frames together. The application uses the presentation time stamp (PTS) from MPEG-2 as their main reference. The application which comes with a hardware box also synchronizes the audio with the video using a hardware system clock to synchronize the signals. The white paper restates the usefullness of timestamps but use hardware with software to accomplish synchronization. The system does not provide solutions about the synchronization problems associated

with processed signals.

2.2 Academic Research

The Audio Synchronization Concept by Michael Robin[6] gives an overview of the audio synchronization requirements in a tele-production studio. The work presented in [6] provides an explanation of the hardware requirements for synchronizing audio and video samples. The approach and methodology involve using a hardware approach using universal clock to synchronize the different signals. For synchronization, Robin[6] necessitates a central synchronizing generator to feed each audio signal with a reference sampling rate. For this to take effect, there would be a separate synchronization socket with every piece of equipment, either audio or video. The main idea is to keep the audio samples in phase with the reference with a tolerance of 5% of the audio packet at the transmitter output and a tolerance of 25% of the audio frame at the receiver input. Consider the sampling frequency of the audio fs and the duration of an audio frame $TF = 1/fs$, therefore, we can calculate the number of audio samples per video frame which is given by:

$$Audio\ Samples\ per\ video\ frame = \frac{Video\ frame\ duration}{Audio\ frame\ duration}\ .$$

This phase relationship between audio and video signal has to be maintained to an integer value so that it is possible to synchronize the audio packets to the relevant video frame. Robin explains that the video samples are synchronized with the universal clock using the vertical blanks in the video as well as the colour black in the signal. Robin also explains that there is a need to re-synchronize the samples from the audio and video when switching of the signals occurs.

Robin's discussion of the synchronization problem is from a hardware point of view. Robin explains the need of a universal clock which is the general solution

in solving the synchronization problem. However, the processing of the signals are only limited to sampling, and no per frame video processing is performed, e.g., edge detection, or further processing on the audio, e.g., speech recognition. The technical report in [6] is also is focused on hardware solutions and not on a software based system which is the case in this proposed project.

Lienhart et al.[7] describe a universal method for distributed audio-video capture on certain devices such as laptops, PDAs, cellular phones, audio recorders and camcorders. The method presented is implemented through a setup and an algorithm that provide synchronization between audio and video for a network that is distributed on multi-channel audio sensors. The objective in [7] is to synchronize the sampling of video and audio without using direct signaling. The sensors are connected to a general purpose computing platform. Lienhart et al.[7] explain that the universal solution provided can also be used to synchronize video streams when respective audio streams are recorded synchronously.

The universal synchronization scheme Lienhart[7] presented is to insert system time-stamps into the audio data at A/D conversion time and process the audio data along with the time-stamp information. This is done by using a dedicated audio channel for distributing the global synchronization information (time-stamps). The synchronized signals are formed in an external master unit with its own clock to modulate an audio carrier signal. These signals are delivered to the platform using dedicated links with little latency (such as a wireless analog FM radio transmitter). The requirement for this solution is an external RF modem and an additional audio input dedicated to the audio sync (time-stamp) signal. RF is used to reduce propagation time and reduce delays. The sync signals are processed in dedicated circuits and delivered by electromagnetic waves. The timing information would then be used by the processing system to convert the sampling rates such that they are synchronized

with the video sampling rates.

The method used by Lienhart[7] is useful in developing systems that use time-stamp information. It presents the idea of using the timing information to edit the sampling period. The system set-up in [7] is vastly different from the one proposed in this thesis. It uses multiple audio channels and computer platforms for capture while in the case presented in this thesis uses only one computer and one microphone. Also the processing of audio and video are more complicated. The only processing in question in [7] is sampling and sampling rate conversion.

VIDAS[8] stands for Video Assisted Audio Coding and Representation. The aim of the VIDAS[8] project is to develop facial animation using information from both audio and video signals. The main components of the project are: facial feature extraction, speech analysis for automatic association of lip movements to phonemes, conversion of video and speech information into MPEG-4 facial parameters, development of 3D head models using MPEG-4, calibration tools for MPEG-4 and animation tools. This system can be used in generating synthetic video with animated voices. The audio is synchronized with the video using information extracted from the lip movements of the original video. This is an example of how some engineers use the video information such as lip movements to synchronize the audio associated with those lips. This is just one example of such work. This work is not useful for the proposed system as one of the objectives is not to use any video object information in the synchronization process. The methods involving synchronization using video information have been researched in literature and therefore, a new method using temporal information is proposed in this thesis. Apart from the clear drawback of occlusion in the synchronization methods involving lip movements, extra video processing would slow the system down. One objective of the proposed system is to develop a fast system and so the decision was not to include any further video processing.

A synchronization method using facial information is presented by Malcolm Slaney and Michele Covell Slaney et al. [9]. The method measures the level of audio-video synchronization by utilizing the facial information from the video. Slaney et al. present an algorithm (FaceSync) which is similar to a Weiner filter by that it combines information from all the pixels to measure the degree of audio-video synchronization. The algorithm uses a face recognition algorithm and canonical correlation to measure the level of synchronization between the audio and video signals. Canonical correlation is a procedure for assessing the relationship between two sets of variables. Slaney et al. [9] describe two steps of the algorithm: the first step is training or building the canonical correlation model and the second step is evaluating or testing the model on the data. A neural network is used to build and test the models. In the training stage the algorithm maximizes the cross correlation between the aligned face image and the corresponding audio signal, while in testing the correlation between the new audio and aligned face (face localization performed on the video) is evaluated. The goal of the FaceSync algorithm is to achieve a measure of correlation between the face of the video object and the corresponding audio and in turn use that information to evaluate the degree of synchronization between audio and video signals. Again this method shows an example of a synchronization approach using the facial features present in the video and correlating it with the audio that is to be synchronized. The approach relates the audio and video to achieve synchronization but uses the video object information to do so.

Chen et al.[10] present the synchronization problem in multimedia applications and categorize two types of synchronization techniques: intra-media and inter-media. Intra-media involves the playback of the medium involved in continuous time. Inter-media synchronization involves determining the scheduling of playback of the medium. These two types of synchronization are usually labeled continuous and discrete. Chen

et al. claim that a synchronization scheme is successful if it performs the inter and intra media synchronization within the precision limits which are dependent on the media application. They propose a synchronization system that performs intra-media synchronization on the media types separately and inter-media on the overall system, which include governing the system clock and communication between the media streams (integration of the signals). Chen et al.[10] use different threads to control different processes in a video-audio multimedia application. One thread (the control thread) is responsible for obtaining the timing information of the video stream as well as controlling the playback. The child thread would control the intra synchronization duties, which involve synchronizing the audio with itself, and the video with itself (e.g., ordering and buffering the video frames). Chen et al.[10] use a time-axis approach which involves synchronizing the media streams to one clock to achieve absolute synchronization. They also use a priority scheme to determine which stream is more critical in the synchronization process, for example, dropping video frames if necessary. The approach was analyzed according to CPU usage and the number of frames to be dropped to achieve synchronization. The buffer system of the media and the synchronization thread allow the integration of separate signals. This system, however, does not include processing models and their relative synchronization.

Huang et al.[11] present another approach to solve the inter and intra synchronization problem. The application discussed uses presentation slides, audio and video frames. They use Dynamic Extended Finite State Machines (DEFSM) to govern their integration process. The DEFSM contains the delays of the processes, the priorities and the decisions based on the states of the DEFSM. Two DEFSM's are used in the synchronization approach, one for the synchronizer, and the other for the actor. The actor is responsible for obtaining the data (capture) and therefore, intra-media synchronization. The synchronizer is responsible for inter synchronization, and, therefore,

the communication between the different actors. The system described runs over a distributed network and therefore, communication between the synchronizer and the actors are done over a communication channel. The synchronizer uses different states to describe different actions and synchronization points in the presentation. It must be noted that the states of the actor DEFSM must be known prior to launching this application. The system presented by Huang et al.[11] provide a synchronization solution to an interactive presentation application however, some details of the media scenario must be known. The use of FSMs in general facilitate the integration of the application, even though the FSMs may be complicated in their state structure.

Kim et al.[12] explore the problem of intra-stream synchronization across computing platforms. The problem arises when the clocks of different platforms are not synchronized. The communication between the streams is delayed due to increased processing of the video stream. Kim et al.[12] use a Time-triggered Message-triggered Object programming platform to control the different threads involved. The object contains methods to control events based on timing criteria. The objective in this paper is to minimize video play back jitter when transmitted and played across computer platforms. Kim et al. describe a target play time (TPT) which is the time of capture plus the streaming delay. In their approach, if the current time is greater than the TPT of the transmitted video frame then that frame is played immediately. If the TPT is greater than the current time, the video frame is buffered and played when the times are equal. The approach is intended to keep the same delay between the captured video in the transmitting platform and playing the video in the receiving platform. This approach uses a global time stamp method to achieve intra-synchronization, but does not give a solution for relating different media types and inter-synchronization.

Qian et al.[13] present a system to enable developing multimedia services over a wide range of telecommunication networks. One objective of the overall architecture is to encompass the problem of inter-media synchronization (between audio and video). The authors present an architecture called TOMA for embedding open distributed multimedia applications. The architecture includes authoring, presentation and operating system functionality. The authors describe the general architecture of their system and describe its objectives without explaining the details of operation. The paper does show, however, the importance of developing a multimedia system with an inter-media synchronization capability.

Splawski[14] explores the mechanisms involved in inter and intra media synchronization. The main methods involve using time-stamps and signalling according to these time stamps, which Splawski calls synchronization instants. Examples of synchronization instants are when the multimedia object begins to be processed, or when a certain duration of time of that object has passed. Splwaski presents different criteria for inter and intra stream synchronization relative to the defined synchronization instants. The solution provided is viable but the synchronization instants are application dependent. Even though the system used is outdated, the work shows the importance of the temporal information of the media stream.

Tasaka et al.[1] tackle the problem of varied network speed to the synchronization problem. Multimedia broadcasted over the internet can suffer from fluctuating network speed which affects the synchronization of the media stream. Tasaka et al.[1] propose solutions to different network problems, and one of them being media synchronization which is the problem of preserving the media's temporal information. Taska et al. use a multi-stream approach. A multi-stream approach means that several media streams are used in the system. This is a similar case to this thesis as video and audio streams are used. In [1], the output time from the source of the first

media unit (MU) is determined and the destination waits for the arrival of the first MU of each stream (in this case video MU and audio MU). The system then chooses the arrival time of the latest stream and uses that time-stamp as the ideal target output time. The ideal target output time is changed into the target output time by adding a maximum allowable delay (caused by network delays). If the delay is within predefined limits then a new output time is derived. The objective here is to keep the delay constant across the MUs transmitted across the network. The solution in [1] uses temporal information to synchronize the media streams, however, the solution is across a telecommunication network and does not account for processing delays of the actual MUs.

Boukerche et al.[15] propose a synchronization system for multimedia streamed over a wireless cellular network. Even though the scope of this thesis does not encompass wireless networks, it is important to highlight that Boukerche et al. use timing information to achieve synchronization. This is by time-stamping the media as it leaves one area of coverage and enters the next and by monitoring the delay times of each MU.

Benslimane[16] explores multimedia synchronization issues across telecommunication networks. Benslimane proposes two strategies for synchronization, one for a network with little or no jitter and the other with unbound jitter. The method explored here from Bensilimane's paper was the model that included the network delay as that would model the processing delay involved in the system proposed in this thesis. Benslimane uses a buffer to store the delayed MUs and the receiver informs the transmitter of the new delay in the network. The buffer size is dependent on the average delay of the network. Benslimane propose a variable buffer size to overcome the varied jitter problem. The buffer size is varied using the new delay that the receiver feeds this information back to the transmitter. With this information the transmitter

decides whether to flush or not flush the buffer and by how much. Benslimane provides a good approach by using a buffer system for integration and synchronization that is dependent on the delays of the network. The system described in [16] provides synchronization solutions across telecommunication networks and does not account for delays involving the processing of the MUs. The MUs in [16] are not processed and so processing delays of the signals are not accounted for. Only network delays are accounted for and , therefore, there is room for improving the system provided in [16].

Lee et al.[17] present a multimedia synchronization scheme that is related to the Quality of Service (QoS) of an application. Lee et al. describe previous methods of multimedia synchronization. These methods include the use of petri-nets to model the concurrent multimedia systems. A petri net is a mathematical and graphical modelling tool that describe the transitions and states of a system. They are mainly used to model networks carrying packets which is the case in Lee et al.'s paper, where they try to improve synchronization over a network. Lee et al.'s method involves using a DTSM model. A dynamic timed synchronization model (DTSM) is used to describe the time medium and the QoS requirements flexibly and scalable. The DTSM models synchronization events such as priority scheduling and timing events, and so Lee et al. expanded the methodology to develop a new algorithm. The new algorithm uses DTSM to extract the timing parameters and base decisions on these properties. The paper in [17] illustrates that the synchronization problem can be solved using different models and can be used to measure the QoS of a network. This comes to show the importance of synchronization in multimedia across telecommunication networks.

Stocia et al.[18] present a synchronization algorithm that uses the audio time to synchronize with the video. Stocia et al. state that using the information from the audio time provide a simpler solution to the synchronization problem than the use of

Petri Nets as used by Lee et al.[17]. Stocia et al.'s algorithm determines the time of audio plus its delay and determines if the video frame to be displayed has the same timing parameter. Basically, the algorithm synchronizes the audio and the video streams using the timing information of the audio playback as it is assumed to be continuous. Stocia el al. iterate the importance of using the timing information to solve the synchronization problem but do not use any processing on either stream and the delay of the system is just due to playback.

Wang et al.[19] present a timing model for an HDTV encoder. They explain that the time stamps encoded in MPEG-2 are retrieved by the decoder to synchronize the output. They highlight the importance of correct extraction of the time-stamps to ensure correct decoding of the stream. The time-stamps are also used to order the different frames. These time stamps are called Presentation Time Stamps (PTS) and Decoding Time Stamps (DTS). These time stamps specify when the frames are to be decoded and presented at the decoder. The PTS and DTS times are also augmented with a delay which represents the buffer delay to accurately represent the timing information of the data. This is another example of the importance of time-stamps in the synchronization of multimedia.

D.Lee et al.[20] present a scheme for synchronization of video frames encoded in the MPEG-4 standard. The synchronization method only uses DTS (no PTS) and applies an adaptive mechanism to minimize the QoS problem[20]. The DTS is used in both the decoded video and audio and is stamped every 100ms. The displayed time is the DTS plus the delay of the decoder. The operating system ensures that the DTS of the video and audio are synchronized such that the presentation is also synchronized. D.Lee et al. illustrate the need to accommodate the delay time of the decoder to establish synchronization, even in a high speed presentation, but again fail to mention any further processing on the decoded audio and video.

Lopes et al.[21] explain the use of MPEG-7 and synchronization in solving the data retrieval and management problem. Lopes at el. look at personalized TV services and video-based surveillance as examples in MPEG-7 because of the multiple stream management that is involved. The MPEG-7 standard has Multimedia Descriptor schemes (MDS) which provide a wide range of description schemes (DSs) for a wide range of applications making MPEG-7 useful in developing surveillance applications. It is also possible to design an application-specific DSs. In [21], Lopes et al. describe a project named RETRIEVE, that is a video surveillance CCTV system that uses MPEG-7. RETRIEVE uses a variety of networked cameras and the challenge lies in retrieving the useful information from them and relating them to one another. They use MPEG-7 descriptors and process the information from the scenes captured. Instead on relying solely on the time-stamps of the video, which is available in MPEG-7, they use the descriptors which provide a summary of the video contents. With this, they claim that more flexibility is achieved when communication with other devices because time is already embedded in the descriptor. One drawback for using the descriptors is that they have to be designed. If the system is meant to be operated as an on-line system, then the number of events to be programmed can be very large. Therefore, it is impractical to program every single event. This means that the success of system in [21] depends on the descriptor. The system in [21] is, therefore, context dependent. Also, in [21], there is no method for synchronization between audio and video MPEG-7. Even though it is possible to describe the audio using MDS, a DSs must be designed to relate the video to the audio. The design of the DSs to relate the audio and video media streams puts a limitation on the adaptability of an MPEG-7 based synchronization approach. Even though MPEG-7 does have time-stamping capability, there is no assurance that their integrity is kept intact after processing of the respective media streams. This does not take away from the usefullness of

MPEG-7 in data retrieval and media stream management.

2.3 Summary

This chapter provided a summary of the important work relevant to this thesis and mainly the synchronization concept. Robin[6] display the importance of using a central clock and maintaining a constant delay and synchronization threshold. Lienhart et al.[7] uses timestamps to coordinate the multiple audio sensors connected to the system. Chen et al.[10] define the synchronization problem of inter and intra media synchronization, while Huang et al.[11] uses finite state machine to solve it. Kim et al.[12] uses threads in a programming environment to solve the synchronization problem by controlling the different media streams along with the use of time stamps. Stradis et al.[5] explain the use of buffers and time-stamps to achieve their objective while Benslimane[16] uses a buffer, time stamps and the delay times to ensure correct synchronization at the output.

The main ideas presented in the related work highlight the use of the timing information of the media streams and to use a buffer, if necessary, to compensate for the delay in processing. One main aspect of this thesis that was not present in the reviewed work, is the accommodation of complex processing on the media and its integration in the whole system. There was no work found on synchronizing and integrating media that involved speech recognition on the audio or object based processing on the video. The only form of processing was present in [21] where MPEG-7 descriptors were extracted to be used in the RETRIEVE project. Furthermore, there was no clear indication that the descriptors would keep their integrity after processing the media streams. Using MPEG-7 as a base tool for synchronization is complicated as descriptors have to be designed to accommodate the processing as well as the synchronization. MPEG-7 is not meant for synchronization of audio and

video streams but rather solves data management problems.

The work presented in the remainder of the thesis establishes a method to synchronize media streams that involve large amounts of processing, using time-stamps, a buffer system to integrate the streams and using the processing delay information.

Chapter 3

Review: Speech Recognition

The system proposed in this thesis requires a method to extract the speech from the speaker. The speech is extracted using a speech recognizer. The part of the system proposed in this chapter describes a speech recognition module. The speech recognition module is responsible for capturing the audio from the speakers, and recognizing the speakers' speech. The recognition would result in the recognized text which in turn is synchronously displayed along with the processed video.

Research on speech recognition has been present since the early 70's [22] or even earlier. The theory has been developed over the years and now with fast computers the results can be realized. However, since it is a large research area, free recognizers are not available. This is due to the amount of work that is carried out in order to develop a recognizer. This work involves preparing vocabularies, grammars, dictionaries, training data and so on. There are commercial recognizers, however, they are not open source and hence not useful for this thesis.

This chapter outlines the motivation and main principles behind speech recognition, as well as the different approaches used. The description of the method used in the system and a description of the implementation is provided in Appendix A.

3.1 Speech Recognition Review: HMM

Hidden Markov Models (HMM) are used to develop the speech recognition module. There are other ways to develop a recognizer such as using artificial neural networks, but it has been discussed in literature that HMMs are the most favored approach.

Burchard et al.[23] propose a phoneme based recognition system using HMMs. Burchard et al.[23] use HMMs because the model can be easily adapted to represent different dialects in speech and hence provide a more robust system. The recognition system is trained by extracting the features of the speech and comparing it to a codebook or dictionary of phonemes. The HMM parameters are then estimated using this information.

Takiguchi et al.[24] use HMMs to adapt the recognition system to distant moving speakers. This is another example of where HMMs are adaptable to be used to develop different purpose recognition systems. The HMM is used to model the acoustic transfer function of the speech captured at different distances. The states of the HMM are uniquely estimated and identify whether or not the sound source is from a noisy environment or not. The adaptability of the HMM to be able to estimate parameters for a noisy environment is the reason behind the choice for using HMMs in [24].

Luo et al.[25] use HMMs to develop a large vocabulary continuous speech recognition (LVCSR) system . The authors develop an algorithm based on HMM to develop state-tying in HMMs such that LVCSR system can be achieved.

Yoshizawa et al.[26] propose an HMM system for word recognition. Word recognition is hard to develop due to the high computational costs associated with it. It is discussed later that word recognition was not suitable for this project. However, it is the work presented by Yoshizawa et al. show that HMMs can be used to develop word recognition systems as well as phoneme recognition systems. The system proposed is

a hardware implementation and therefore, the system is faster to accommodate the word recognition.

It is clear that HMMs are a popular choice for developing speech recognition systems. The details of the actual theory of HMMs are presented in the remainder of this chapter.

3.2 Speech Recognition Theory

Speech recognition is the process of transcribing speech into text. A speech recognizer is used to perform this task. The speech recognizer usually contains a vocabulary or grammer to refer to when transcribing the speech. In most speech recognition systems, the recognizer would split up the speech into phonemes. Phonemes are pronunciation or linguistic units [27]. The recognizer would then match the phonemes from the actual speech to the phonemes in the grammer and produce the transcribed word.

To develop a speech recognition system a mathematical formulation of the problem should be developed. This is discussed in the following section.

3.2.1 Statistical Background on Speech Recognition

This section describes the basic mathematical concepts behind the speech recognition problem. By understanding the mathematical principals behind speech recognition, the process of implementing the system becomes clearer.

First, lets denote the acoustic or speech information to be recognized as A. The set of A is a set of digital symbols, since the raw acoustic information is digitized to perform the recognition. Therefore, A can be considered as a set of symbols taken from a larger set $\mathcal{A}$[27],

$$A = a_1, a_2, ..., a_m \quad a_i \in \mathcal{A}. \tag{3.1}$$

The sequence of symbols a_i are generated with respect to time in accordance with i[27]. Let W represent a string of n words that are defined in a vocabulary $\mathcal{V}$,

$$W = w_1, w_2, ..., w_n \quad w_i \in \mathcal{V}. \tag{3.2}$$

$P(W \mid A)$ is the probability that the sequence of words W was spoken in A. The recognizer is expected then to choose the most likely word string in A which is represented as $\hat{W}$, as in the following:

$$\hat{W} = arg \max_{W} P(W \mid A). \tag{3.3}$$

From Bayes rule, $P(W \mid A)$ can be rewritten as [27],

$$P(W \mid A) = \frac{P(W)P(A \mid W)}{P(A)}, \tag{3.4}$$

where $P(W)$ is the probability that the string of words W is uttered and $P(A \mid W)$ is the probability that when W is uttered, the acoustic information A is available. $P(A)$ is the average probability of the acoustic information being observed.

Now since $P(A)$ is fixed, the speech recognizer goal is to find the word string $\hat{W}$ that maximizes the product of the probabilities. The following equation [27] describes the main requirements of a speech recognition system,

$$\hat{W} = arg \max_{W} P(W)P(A \mid W). \tag{3.5}$$

3.3 Components of a Speech Recognition System

The main components of a speech recognition system can be described using Eq. 3.5. The first task is to extract the information needed from the acoustic data A. That means a method must be developed to extract the symbols a_i from A as in Eq. 3.1. Therefore, the task is to capture the audio using a microphone, which the hardware transforms it into an electrical signal, then sampling that signal and then processing the samples to extract the useful information.

Speech signal processing is the preparation of the raw speech signal in a form that can be used in speech applications. The information from speech can only be obtained in the form of a speech waveform[28]. The speech waveform is digitized in two fundamental steps. The first step is to sample the signal which is the process of converting a continuous signal (in this case the raw acoustic speech waveform) into a discrete signal. The second step is to quantize the discrete signal, which means assigning a value from a code set to the variable discrete value.

After quantization, the symbols generated are clustered in groups, called vectors[27]. This process is called vector quantization. The general idea of clustering the symbols gathered by quantization is that speech can be represented in groups of phonemes. Phonemes is the pronunciation specification of words [27]. Since phonemes can be grouped to form words, the symbols generated by quantization can be grouped to form clusters. These clusters which represent a N dimensional characteristic space for the phones have a center (center of gravity). The center of the space can be used as a good approximation of the actual symbol. Vector quantization is a method to determine the center of the aforementioned clusters and therefore, provide the framework to describe the phonemes.

Equation 3.5 also requires the probability of when the speaker uttered the word sequence W from the acquired speech A. Specifically, $P(A|W)$ needs to be deter-

mined. Since the number of probabilities of all possible combinations of A and W is too large for a lookup implementation, a statistical acoustic model of the utterance is required.

The most common statistical model used is the Hidden Markov Model (HMM). Other methods do exist, like methods based on artificial neural networks and dynamic time warping. The speech recognition method employed in this project, uses HMMs.

3.4 Hidden Markov Models (HMM)

Markov chains are used to model real-time statistical events using the probability of occurrence, which is based on the near past [29]. From this definition, real events can be modeled by first order Markov process. A first order Markov process is when the probability of an event is based only on the preceding state of the model.

3.4.1 Markov Chains

HMMs are applied in several analysis tasks such as deciphering cryptograms, to solving problems relating to the field of bioinformatics. Similarly, the HMM model is used to solve the speech recognition problem.

To describe Markov chains, consider a sequence of random variables $S_1, S_2, ...S_K...$ which take their values from a finite set, $\mathcal{S} = 1, 2, ..., c$. With Baye's rule it follows that:

$$P(S_1, S_2, ..., S_K) = \prod_{i=0}^{n} P(S_i|S_1, S_2, ..., S_{i-1}). \tag{3.6}$$

The random variables in Eq. 3.6 form a Markov Chain [29]. A Markov chain is a discrte-time stochastic process that follow the Markov property[29]. A Markov chain is used to describe the states of the system and their transitions. The Markov

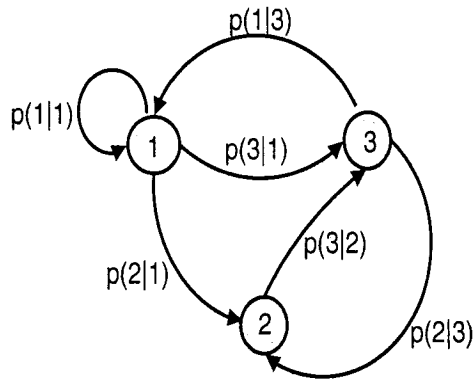


Figure 3.1: 3-State HMM.

property implies that the system is memoryless, which means that the transition of the next state solely depend on the current state and not on any previous state. By this Eq. 3.6 becomes

$$P(S_1, S_2, ..., S_K) = \prod_{i=0}^{n} P(S_i|S_{i-1}). \tag{3.7}$$

Therefore, the random process described in Eq. 3.7 [29] means that the variables at time i depend only on the previous time $i-1$ and not on any other time prior to that. This means that the system is time-invariant which is a property of Markov chains.

The states of the random process, which is modeled using a Markov chain, are represented by S_i. The following equation represents the state transition as a function:

$$P(S_i = x^{'}|S_{i-1} = x) = p(x^{'}|x) \quad \forall\ x, x^{'}\ \epsilon\ \mathcal{X}. \tag{3.8}$$

Equation 3.8 [29] evaluates the transition function $p(x^{'}|x)$. If the set of $x, x^{'}$ is finite, the random process can be represented as a sequence of states with their transition probabilities defined.

Figure 3.1 illustrates a finite set of states with their transition probabilities defined. The figure shows a three state Markov Process [27], and the probabilities $p(\cdot \mid \cdot)$ are the transition probabilities.

3.4.2 HMM concept

The Markov model discussed so far defines each state in the chain with an observable event. This model is proven to be too restrictive to use in modelling actual events. The Markov model concept needs to be extended to include observational data that is a probabilistic function of the state of the model. This model is a double stochastic process, where one stochastic process is hidden and can only be observed through the other stochastic process that produces the sequence of observations [30]. This model is dubbed as the Hidden Markov Model (HMM).

An HMM is defined by the following[30]:

- The number of states in the model, N. As explained earlier, the states are hidden, however, there is a physical definition to each state. For example, if the urn and coloured ball problem is to be modeled using HMMs, the states would represent the urns. Each state is denoted as $S = \{S_1, S_2, ..., S_N\}$.

- The number of observation symbols M per state. Observation symbols is the output of the system that is being modeled. For example, in the urn and ball model, the colour of the ball picked from the urn is the observed symbol, with the state defined as the urn it was picked from. The actual observed symbols are written as $V = \{v_1, v_2, ..., v_M\}$.

- The state transition probability. This probability determines how the states in the model are changed or how the model can move from one state to another. The state transition probability is denoted as $A = \{a_{ij}\}$ where i, j are the source

and destination states. In the urn and coloured ball problem the transitions are the probabilities of any other ball to be picked.

$$a_{ij} = P(S_j|S_i) \quad i \geq 1, \ j \leq N. \tag{3.9}$$

- The probability distribution of the symbol observed in j.

- The initial state distribution.

To build an HMM and to deem it usefull a model has to be developed to predict future events based on the already available data. The available data is dubbed the training data. The first task is to train the HMM. This means to optimize the HMM parameters (transition and output production probabilities) to best fit the given observation data (training data) on which the HMM is based on. The second task is to uncover the hidden model which means to estimate the most accurate state transition sequence in terms of the observed data. An optimality criterion is used to solve this problem. The last task is to test the HMM and its capability to determine the output of unknown data (testing).

If the speech recognizer is to be designed for isolated word recognition (simple recognizer), the first task would be to build individual word models. The word models that would be used to train the HMM, would be based on the spectral vectors obtained from a spectral code book. The spectral vectors, which make up the word, is the training data used to build the HMM. The first task in this case would be to estimate the word models from the acoustic information and the initial parameters of the HMM. The second task would be to assign the states of the HMM to the word models and its sequence. The third task is to test the recognition capability of the HMM designed [30].

Phonetic Acoustic Model

As an example, the construction of an HMM to model a phonetic based recognizer is described. The recognizer that is used in the system is a phoneme recognizer. The main objective, as discussed earlier and from [27], is to maximize the probability that the word W is uttered in the A, to obtain $\hat{W}$. The objective was defined in Eq. 3.5.

To model a word string W (Eq. 3.2), and its corresponding individual words w_i, it is clear that they are made up from smaller blocks. The aim is to model the smaller building blocks in HMMs and to concatenate them to define the word string W. Separating the model into smaller building blocks is useful in speech recognition since vocabularies can be in the range of tens of thousands, and in turn very difficult and almost impossible to model.

The phonetic acoustic model is based on an intuitive linguistic concept [27] and there is a method to split up vocabularies into smaller building blocks. The HMMs for words, described using phonemes is constructed as follows:

1. A phonetic dictionary is created for the required vocabulary. In symbols, a relationship between each word v and a sequence of phonemes $\Phi(v) = \varphi_1, \varphi_2, ..., \varphi_v$ is created. $\Phi(v)$ is the phonetic pronunciation of the word v.

2. For every different phoneme used to define the vocabulary, an elementary HMM is used to model it, with starting and ending states.

3. The HMM for a word v is just a concatenation of the HMMs of the phonemes.

4. As for a string of words, W, the HMM is defined as another concatenation, but of the HMMs of the words v that make up the string. HMMs for silence symbols are inserted if required.

5. To estimate the parameters, the Baum-Welch algorithm is used [27]. This is the training process that was mentioned previously.

3.4.3 HMM Limitations

From the definition of first-order Markov chains in Sec. 3.4, HMM states depend only on the previous state, and this may limit HMMs in speech applications. The observation output is dependent on the state that generated them and not on any neighboring output. In speech, the current phoneme is highly dependent on the context. This means that the recognized phoneme may depend on the previous states, and not just the preceding one. It is possible to develop higher order HMMs so that dependence is not only on the previous state. However, this makes the development of the HMM more complex, both in implementation and estimation of its parameters.

Furthermore, HMMs are only well defined for processes that are a function of a single independent variable such as time or a one dimensional position. It is practically impossible to define HMMs for more than a single independent variable. Therefore, it would be hard to combine the HMM model for speech recognition with that of another model [30]. However, HMMs still remain to be the best statistical method in solving the speech recognition problem due to the modelling structure of the phonemes.

3.5 Feature Extraction

Feature extraction is the process of obtaining the important information to uniquely identify the characteristics of the speech waveform. Basically, the raw speech signals, for example, X, are processed to extract significant data for recognition, $Y = g(X)$ [31].

The most widely used analysis method in automatic speech recognition uses the Mel-scale cepstrum. The cepstrum, comes from cepstral analysis, and it is the process of modelling speech production. The cepstrum is a different way of modelling the spectrum of the speech signal such that it is proven to more accurately represent the

human hearing and vocal tract system. The cepstrum is taken by the inverse Fourier transform of the complex logarithm of the discrete speech signal $x[n]$, as in

$$\widehat{x}[n] = \mathcal{F}^{-1}\{logX(e^{j\omega})\}. \tag{3.10}$$

The logarithm of the speech spectrum is a characteristic of the human hearing system and it also reduces the amplitude component at every frequency in it [31]. Futhermore, for practical applications, the real part of the cepstrum is only needed as in

$$\widehat{x}_r[n] = \mathcal{F}^{-1}\{log|X(e^{j\omega})|\}. \tag{3.11}$$

Now to make it the cepstrum applicable in digital algorithms as is the case in speech recognition, Eq. 3.11 is used with the discrete Fourier transform, (DFT), in place of the Fourier transform.

To use the cepstrum, the Mel-scale is used to represent the coefficients. This is called the Mel Frequency Cepstral Coefficients (MFCC). The Mel is a scale of pitches, and is usually represented versus a range of frequencies. The reference is set by equating a 1000Hz tone to 1000 Mel's on the scale.

3.5.1 MFCC Processing

The Mel Frequency Cepstral Coefficients (MFCC) is the most widely used feature extraction parameter in speech recognition. This is because MFCC's provide good differentiation and are very flexible in terms of modifications [22]. In brief, to extract the MFCC of the speech signal, the signal is first divided into frames with the aid of a windowing technique. After that, for every frame, the amplitude spectrum is obtained and its logarithm is calculated. The logarithm is then converted to the Mel

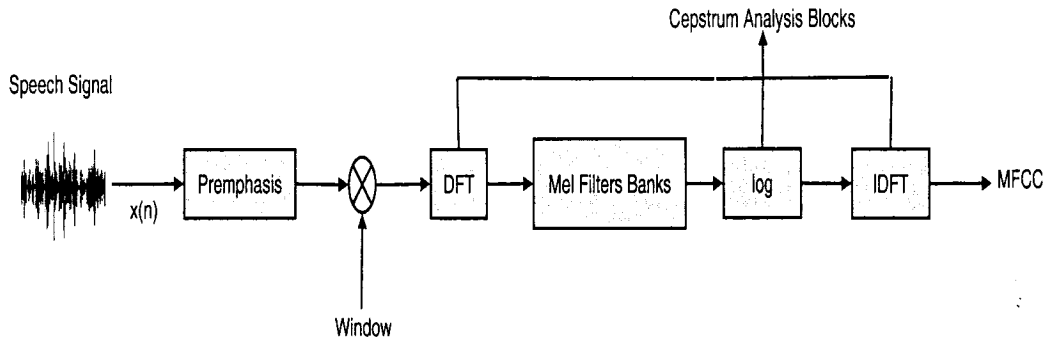


Figure 3.2: Block Diagram of MFCC Processing.

spectrum, and the cosine transform applied. A general technique of extracting the MFCC, and in turn the feature vectors of the speech data, is illustrated in Fig. 3.2 [31].

The *Premphasis* blocks' purpose is to increase the amplitudes of the high frequency components which usually have much lower amplitude levels than the lower frequency ones. The frequency components of the speech signal form the characteristic of the uttered phoneme and therefore, it is important to represent all the frequencies with similar amplitudes. This process is usually done using a first order FIR filter. Noise cancelling and silence detection is also performed in this step of feature extraction.

Following the premphasis block, a *Window* is applied to the signal. Windowing a signal is used to split the signal up into frames. In this case, the speech signal is windowed to form frames, where the statistical characteristics within the frames are invariant with respect to time. If the speech signal was already invariant there would have been no need to window the signal.

The next processing step is to take the *DFT* of the windowed signal. Since the standard spectral analysis depends on the frequency components of the signal and in turn the Fourier transform, the DFT acts to reduce the complexity of the analysis. The reason is that evaluations are only performed at discrete values of the frequency.

To further reduce the complexity, the Fast Fourier Transform (FFT) is used. It is important to note that the phase information of the DFT is discarded as the phase information does not provide any useful information in the analysis.

The *Mel Filter banks* is then applied to the output of the DFT (or the FFT in implementation). The filter banks are used to extract the frequency components from the whole spectrum. Usually a set of 24 filter banks are used because it conforms with the human ear processing system. The filters, used to extract the components, are spread out non-uniformly along the frequency spectrum. The filter banks are more concentrated below the 1kHz [31] mark because the frequency components contained within this range have more information on the vocal tract[31]. The most widely used filter banks in speech recognition are Mel-filter banks. Mel filter banks use the Mel scale, and are uniformly spaced below the 1kHz mark, and are on a logarithmic scale after 1kHz [31].

The $log||$ is then taken to follow the human ear model as well as cepstral analysis (Eq. 3.11). The magnitude discards the phase information contained after the filtering process.

The *IDFT* block is the inverse discrete Fourier transform. This block computes the final Mel Frequency cepstrum computation (MFCC). The inverse reduces to a discrete cosine transform because the log power spectrum output from the previous block has the property of being real and symmetric. The DCT produces highly uncorrelated features [31] which is needed to reduce the number of parameters to be estimated in the HMM and therefore, reduces the complexity in recognition.

3.6 Summary

This chapter provided the background of the concepts behind developing a speech recognizer. The chapter explained the definition and motivation of using HMMs as

the statistical model for speech recognition. The recognizer was implemented using the Hidden Markov Model Toolkit (HTK). The process of setting up the resources, training the HMMs and testing them is also explained in Appendix A. Furthermore, details on how to improve the recognizer is provided in Appendix A.

Chapter 4

Proposed Audio-Video Synchronization

4.1 Motivation

Synchronization between audio and video signals is needed in multimedia applications which involve some form of signal processing. For example, this processing may include recording audio and video signals separately, sound enhancement, and video enhancement. For example, in a recording studio, audio tracks are recorded separately as well as the video recordings and so a method must be developed to re-synchronize them in order to play them back. Without the synchronization the recorded data may be irrelevant to one another in terms of timing and incoherent to the observer.

Multimedia synchronization is defined as maintaining intra-media and inter-media timing relationships so that multimedia data can be processed synchronously within an acceptable range compared to the original timing at the source [17]. Recall from Ch. 2, intra-media involves the playback of the medium involved in continuous time [10]. Inter-media synchronization involves determining the scheduling of playback of the medium. These two types of synchronization are usually labeled continuous

and discrete. The system proposed here performs mainly inter-media synchronization, even though some aspects of intra-media are involved. This is because the synchronization problem lies between the audio and video stream, causing it to be an inter-media synchronization problem. The audio and video streams need to be synchronized at the output.

The synchronization of signals is their alignment with respect to a specified element. In this system, synchronization of audio and video signals is performed with respect to time. The synchronization problem is present whenever there is processing on separate signals. This means that the signals are separated prior to their processing. In a digital video camera, the video and audio are captured at the same time and are coupled and recorded. In fact, the amount of raw processing on either of those signals is minimal and so the signals are synchronized when recorded and in playback.

The integration of an audio-video system is needed when the media streams come from different sources and are processed separately. As soon as the signals are separated, there must be a method to re-synchronize the signals. The streams need to be captured, processed and outputted according to the synchronization information. The integration part of the system is needed to communicate between the different system modules.

If the system was setup such that the capture of audio and video signals was performed using the same device (e.g., a camera with an embedded microphone), the signals would have to be separated to carry out the processing (speech recognition and edge detection). The delay added from both processes are not equal and therefore, a synchronization method is needed to realign the signals with one another.

4.2 Discussion on the Effectiveness of the Synchronization Approaches

In the system proposed in this thesis the audio and video signals are captured using separate sensors. The audio is captured through a microphone which is connected to the sound card of the machine, while the video frames are captured using a web cam which is connected to the USB port of the same machine. The audio captured is used by the speech recognition module while the video frames captured is processed using video processing libraries (VidPro libraries) [32].

A synchronization process may not be needed if there is no incurred delay in the signal processing and that the capture of the signals were at exactly the same rate. This is not the case in the system discussed here. Even if the system was changed such that the capture of audio and video signals was performed using the same device (e.g., a camera with an embedded microphone), the signals would have to be de-coupled and then processed separately. Since the delay incurred from both processes is not uniform, a synchronization method is needed to realign the signal.

There are audio-video synchronization methods in literature that involve the lip movements of the objects in the video frame. Lip movements are tracked on the objects in the frame and related to the audio, and in turn develop a synchronization method. The obstacles in this method include multiple speakers and incoherent lip movements (including occlusion of the lips). If the synchronization method did not depend on the video information then the problem of occlusion can be eliminated. The objective of the proposed system is not to use any video information and solely base the synchronization approach on timing information obtained from the video and audio signals. Furthermore, using extra video processing to determine synchronization will slow the system down. The system is already loaded with complex operations

from the speech recognition to the edge detection. One way to achieve synchronization without adding significant overhead is to use timing information. Timing information is precise and does not involve complex operations to extract.

The proposed method in this thesis supports multiple speakers. The system uses the timing information and therefore, is independent of the number of speakers present. This is one main advantage of using the timing information for synchronization and not any video information. Also note that the system is stream based, i.e., the video and audio are continuously captured and processed.

4.2.1 Processing Delays

The processing delay is incurred from the time the raw signal is available to the time the processed signal and the information obtained from it is available. In terms of the audio, the audio processing delay is incurred from the time the audio is captured to the time the recognized speech (transcribed text) is available. The video processing delay is the time between when the raw frame is captured to the time the contour image (after the binary edge detection) is available. These delays are not uniform and varies according to the amount of data to be processed. In the audio processing, the delay is directly proportional to the length of the utterance, while in terms of the video processing, the delay is directly proportional to the number of objects in the frame.

The delay in the speech recognition is a result of the various processing steps discussed in Ch. 3. These steps are involved in the recognition process. In the online recognition, the recognizer must extract the features, look up the closest matching phonemes and output the results. It is clear that the larger the dictionary of phonemes the longer the time of recognition. In fact, the recognizer was originally built using the HTK tools (Appendix A) to perform word recognition and not phonemes. The

set of resources for word recognition was abandoned because the delay incurred was very large during online recognition. The large delay times is due to the word lookup after the phoneme recognition. The delay times are also large because of the ambient noise present in the room, that cause the recognizer to try to fit the noise to the phonemes and then the words in the dictionary. The large delay times would cause the synchronization approach based on the timing information more difficult or even impractical to achieve. For this reason, the recognizer was set up to perform phoneme recognition, and eliminate word matching. This resulted in faster recognition times and in turn facilitated the realization of the synchronization approach.

The video processing delays were on a per frame basis as opposed to a per utterance basis in the audio delay case. The video frame is captured and passed through a filter sequence until the contour image is obtained from the binary edge detection. The video processing delay is increased with increased movement in the video as well as the number of objects present in the frame. In the implementation, the delay of the speech recognition is larger than that of the video and therefore, the system is implemented to accommodate that. However, the proposed time-stamping approach is also suitable for the circumstance where the video delay is larger than the speech recognition's delay, since all that is used to synchronize the system is the timing information.

4.2.2 Synchronization using Frame Dropping

A frame dropping approach was first discussed as a proposed method for synchronization of this system. Frame dropping is a technique in video playback that increases the video frame rate to improve video and audio quality and maintain an overall rate of playback. In this sense the frame dropping technique is a method for synchronization. Chen et al.[10] proposed a frame dropping technique to synchronize the media

stream.

To increase the frame rate to a specific level, a number of frames can be dropped. The lower the number of frames to be processed the higher the frame rate. Therefore, it is possible to adjust the frame rate such that the video is synchronized with the audio.

The question is what if the audio processing is delaying the video? This means that the audio is what is incurring the delay in the whole playback. It is not possible to drop audio packets as information will be lost since audio is continuous. The human ear is more sensitive to discrepancies in missing speech than the human eye is susceptible to missing frames. It is not useful to drop frames in the case where the audio delay is larger than the video. In fact, in this case you need to slow down the video processing by inserting more frames or repeating the frame, which is more practical.

Another approach must be developed since the audio delay incurred from the speech recognition is much larger than the video delay incurred by the video processing. A frame dropping technique would not be effective.

The proposed approach in this thesis accompanies for the delays incurred in the video and audio processing. The system proposed is designed to eliminate occlusion errors and synchronize the delayed audio with the video frames.

4.3 Proposed Approach: Time Stamping

Time stamping is a method of obtaining the actual time at a specified point in a process. The objective is to be able to extract useful time-stamps from the audio and video processes respectively to be used in the synchronization approach. Chapter 2 discussed the various methods, including time-stamp methods, to solve the synchronization problem. If the time of processing of each video frame is calculated as well as

extracting their time of arrival to the processing module, the actual time of the frame can be established. These times can be thought of PTS (presentation time-stamp) and DTS (decoding time-stamp) respectively, as used in the MPEG decoder. The same reasoning follows for the audio frames/packets. If the actual time of the audio is known via the time-stamps, then the time to insert the audio information (i.e., the recognized text in the case of this system) onto the video is known. If the approach developed is to be able to insert the recognized text information onto the right frame in terms of time, then the system is able to synchronize both processes to within one frame. This approach is described in the following subsections. This proposed approach is valid for both cases, either the video processing delay is larger than the speech recognition delay, or if the speech recognition delay is larger than the video processing delay.

4.3.1 Video Time-Stamps

The video processing on the captured video frames is performed via a C++ program with the aid of C++ libraries, that was developed by the VidPro Research Group at Concordia University [2].

To obtain the real time of video captured, the time of processing each frame is obtained using time-stamps. For each frame, a time-stamp is taken at the moment the raw frame is available from capture, and subsequently another time stamp is taken after all the processing is done. With this information, the delay of the video processing is obtained and the actual frame rate can be calculated. This frame rate reflects the amount of time it takes for one frame to be processed. This frame rate differs than the frame rate if the processing was bypassed. Fig. 4.1 shows the video processing and its respective time-tamps.

The initial time-stamp, "Time-Stamp i", is taken as soon as the frame is captured.

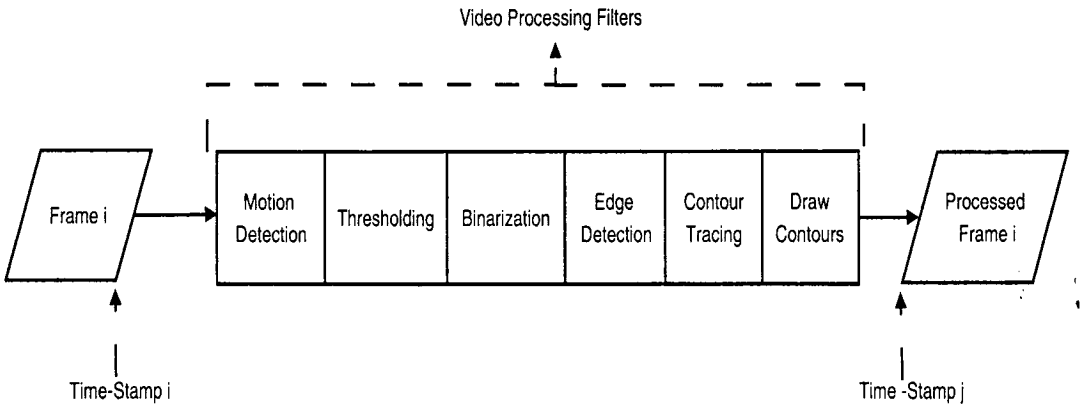


Figure 4.1: Time Stamping Model for Video Processing.

The frames are then processed through the various filters until the contour image is obtained. The second time-stamp, "Time-Stamp j", is taken as soon as the contour image is ready. Simply, the difference between the two time-stamps results in the processing time for one frame as in

$$\tau_{frame} = T_{frame_j} - T_{frame_i}. \tag{4.1}$$

The frame rate, F_R, is calculated by using the number of frames being processed. This is the F_R of the output and not of the capture F_R. If the number of frames is known, the frame rate can be calculated using the total time of processing. In fact, the system keeps track of the number of frames processed, F_{Count}, and uses it as an index to some of the buffers and arrays discussed later in this chapter. The calculation is

$$F_R = \frac{F_{Count}}{\tau_{frames}}. \tag{4.2}$$

The total time of processing τ_{frames} is the accumulated times from Eq. 4.1. This is the processing times which are accumulated to represent the total time of processing

F_{Count} frames. The F_R is a sliding frame rate by that it is calculated using a temporal window (fixed number) of frames. Therefore, the frame rate would represent the recent processing times of the video frames. This is to ensure the frame rate is representative of the most recent timing specification of the video sequence. In turn, F_{Count}, is a constant number that is used to calculate the sliding F_R over the past F_{Count} frames.

With obtaining the times from Eq. 4.1 and Eq. 4.2, the timing information for the video to be used in synchronization is obtained.

4.3.2 Audio Time-Stamps

Obtaining the audio time-stamps involves extracting the time of the utterance and the delay of the recognizer. For the timing information of the audio to be useful in synchronization, it must include the times of the utterances, the silence times between the utterances, and the total time of audio captured. If all the timing information is known about the signals, then no other raw information is needed to synchronize the signals. The challenge lies in extracting and processing the timing information from the audio and speech recognition system in place.

One audio time-stamp is the time of the start and end of the spoken utterance. Another timing parameter is the silence time between the utterances. Silence times is the time where no speech is present. If the time from the audio processing is going to be compared to that of the video processing, the total time of acoustic processing must be considered, including the processing of the silence time. Silence intervals are not processed in this system and are just suppressed. Therefore, a method must be implemented to detect and calculate the silence times. This is explained in Sec. 5.1.

As for the time-stamp of the explicit utterance, this is obtained from the output file of the online recognition. When the recognizer is run online, the output is stored

in a text file. This text file contains the recognized phonemes as well as the times of their utterance. An example is shown below:

```
0 300000 !ENTER -2937.495605
300000 2100000 sil -1417.729736
2100000 2900000 t -680.301270
2900000 3300000 ah -377.549255
3300000 4500000 ch -1053.523682
4500000 6200000 ao -1562.439331
6200000 6800000 el -444.782562
6800000 7600000 s -691.934998
```

The first term is the time it takes for the recognizer to start. This time does not vary between different utterances. The remainder of the entries have their start and end times of the recognized phoneme. For example, the phoneme "ah" which is the fourth entry in the list, has the start time of 290000000ns and an end time of 330000000ns. Note that the time is represented in the file are in units of 100ns. With this information the length of the spoken utterance is known after recognition. It is important to note that this information is not available in true real-time and is obtained after the process of recognition. This output file from recognition, only contains the timing of the utterance and does not include the silence times between utterances and the delay time of the recognizer.

4.4 System Adaptability

The timing information of both media streams extracted from the speech recognition and the video processing enables the scheduling of the output. This section discusses the robustness of the system in terms of variable delays. The implemented system is designed knowing that the speech recognition delay is higher than the video processing delay. This means in the implementation that the video processing thread is waiting for the speech recognition thread to complete processing. In Ch. 5, the

system communication is designed knowing the fact the delay of the recognizer is larger than the video processing. The use of the time-stamps in the synchronization process is explained in Ch. 5.

However, in terms of the theory of the approach, the time-stamping method is independent of which process has the longer delay. The timing information provides everything needed for synchronization, for both cases. If the timing information is gathered and both processes outputs are buffered to some degree (which they are, Sec. 5.4), then the synchronization is successful.

The system proposed is also adaptable to multiple speaking objects. The system is not dependent on the number of objects present. The time-stamp information does not rely on the number of detected objects or the number of speakers in the frame. The only limitation is that the current speech recognition system is only trained for the author's voice, but that can be changed, if required.

4.5 Summary

This chapter explained the synchronization methodology of the proposed system. The approach uses time-stamps to determine the timing information of the video and audio streams. This chapter explained concepts that can be used in multi-stream synchronization as well as inter-stream synchronization. The whole system serves as a synchronized integrated audio-video multimedia system, which is the objective.

The system is also not confined to the audio delay being larger than the video processing delay. Since the system uses the time-stamp information of both media streams, the system is adaptable to accommodate synchronization in both cases where the audio delay is larger than the video delay and the video delay larger than the audio.

Chapter 5

Proposed System Integration

This chapter contains the discussion on the methodology of using the time-stamp information, explained in Sec. 4.3. It starts by discussing the system set-up, followed by the main implementation details of integrating the main system modules described using the aid of flow graphs and block diagrams. Note that the communication between the system is implemented knowing the fact that the speech recognition delay is larger than the video processing delay. However, the general aspects of the system are adaptable to when the video processing delay is larger than the speech recognition delay.

5.1 System Set-up

The full implementation is setup on a Linux Fedora Core 3 machine, using C++ as the programming platform. In the software implementation there are three major modules: the video processing module, the audio processing module and the synchronization module. Each of these modules are run using C++ thread which was implemented using the ZThread C++ library.

Threads in C++ programs enable multiple processes to be run concurrently re-

gardless of their position. A thread is a sequence of instructions that is executed within a program. This means that a thread is just a regular program and each instruction is executed sequentially, and is controlled by the `main()` part of the program. A regular program that does not use threads only has one process (and therefore, one thread). To achieve parallelism in these programs semaphores and different system calls can be used, such as `fork()` and `exec()`. The disadvantage of using system calls to achieve parallelism is that they involve kernel intervention and in turn, incur a significant amount of overhead time during run-time.

In a program that uses threads, different parts of the code are executed using different threads simultaneously. The different threads share variables and structures and can access and alter their contents (unless specifically locked). The disadvantage of using threads is that the programming is more difficult and takes longer than programming a non-threaded application. This is because of the presence of dependencies between the threads. The advantage of using threads is the speed of communication between the threads. In the case of the proposed system, speed is of upt most importance as no extra overhead is desired over that of the video and audio processing delays. Another advantage is if one thread is blocked due to I/O operations in a different thread, the thread simply pauses and the rest of the program continues running. Also threads are highly portable across different Linux platforms. Single processor machines use threads with unnoticeable effect in the speed of processing.

In the system proposed, there are three main modules that are run under one program using three threads. One thread contains the video processing module, a second thread contains the audio processing module which is indirectly linked to the speech recognizer, and the third thread is the synchronization module, which makes use of the timing information from the other two threads to synchronize the output. The audio - video integration is performed by communication between three threads.

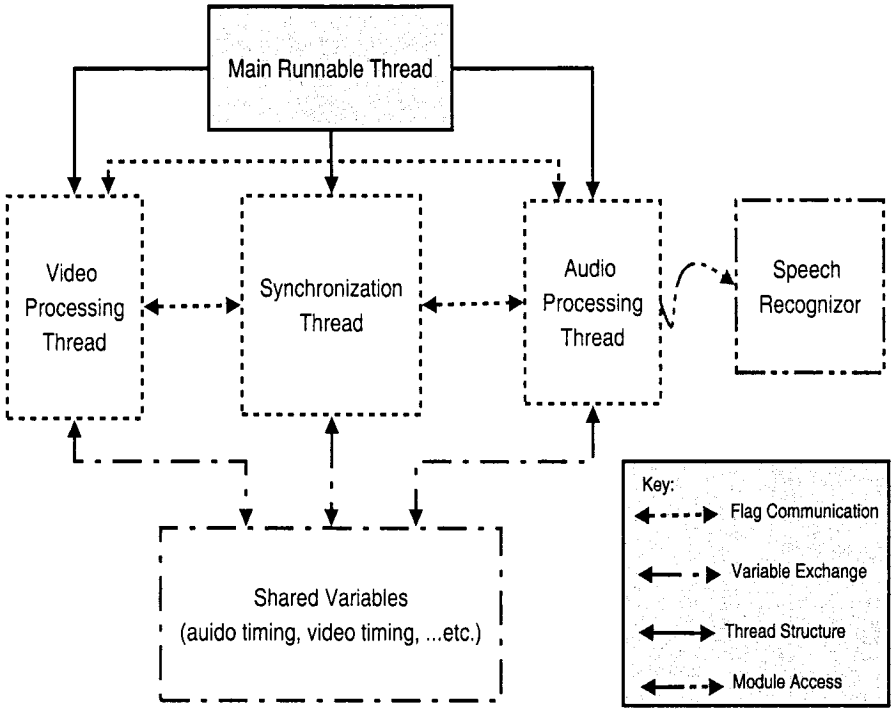


Figure 5.1: Block Diagram of System Integration.

In Fig. 5.1, the audio processing module is split into two parts. The first part is the speech recognizer module which is run using the HTK tools described in Appendix. A. The speech recognizer performs the speech recognition and produces the output file which contains the results of the recognition as well as its respective timing information. The second part is the audio processing thread which processes the output of the recognition by reading in the output text and determines time-stamps that are useful in the synchronization module.

The video processing module is run using the video processing thread and is responsible for capturing the video from a USB camera, performing the necessary format conversions, face detection and then finally the object edge detection. This module also processes the time-stamps and calculates the frame-rate of the processed video.

The synchronization of the system is implemented in the synchronization thread.

This module makes use of the timing information (explained in Ch. 4), from the other threads to implement the synchronization algorithm.

To communicate between the modules and to integrate their respective processes, a flag signalling system is used. A flag system is similar to a finite state machine system as the value of the flag describes the state of the system. In general, the system is implemented using threads and flags and can be thought of as a combination of the system presented in [11] and [12]. The flag system is described in the upcoming sections.

The following sections explain the modules' specifics and describe the flow of data.

5.2 Audio Processing Module

After the resources for speech recognition are developed (i.e., the training process and the adaptation of the speakers voice), the recognizer is ready to be integrated in the full system. The audio processing thread of the system is responsible for making use of the output file from the recognition process.

The first task is to read the phonemes and the relevant time stamps from the recognizers' output file. As seen from Sec. 4.3.2, the output file is in standard format and so reading the file is the same for every output. The challenge lies in extracting the timing information from those files and the recognized phonemes. The output file contains the time-stamps of the phonemes as well as the phonemes themselves. The total time of the utterance is represented by the end time of the last phoneme. The objective in processing the output file is to extract these times (i.e., the first phoneme time, the last phoneme time and all the phonemes themselves).

However, these times are not representative of the times between the recognition of different utterances and silence times (i.e., the recognition delay). If the time-stamp from the output recognition file is the only timing information to be used in

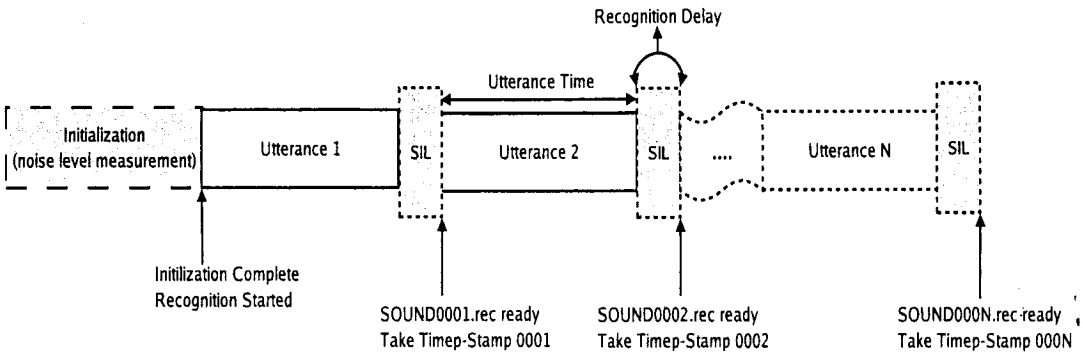


Figure 5.2: Time-Stamps between Output Speech Files.

determining the synchronization with the video frames, then the time between the utterances will be unaccounted for, and the synchronization will be unsuccessful. A method must be implemented to automate the measurement of the time between the utterances as well as the silence times. It must be noted here that the silence times that are represented in the output file at the start and end of the output file, are the recognizers times until the silence is detected. As soon as the silence is detected, the recognizer starts to recognize a new utterance. Therefore, it is important to detect these silence times and eliminate their timing information from the synchronization process, such that it does not affect the time the recognized phonemes are entered onto the frame.

The method implemented in obtaining the times between the utterances, uses time-stamps to record the start and end of the recognition relative to a system clock. These are system time-stamps. The time-stamps are taken once the output file is available to access, which represents a recognized utterance. The number of output files has to be monitored and the time-stamp for each file is recorded. The times between the system time-stamps and the utterance time represents the recognizer's delay in recognizing the first utterance. This approach is illustarted in Fig. 5.2.

The time of the utterance is known from the output file itself. The time between

the utterances (the recognizer delay) is calculated by the difference between the time of the utterance (obtained from the output file) and the time-stamp of the file (system time-stamp00N). With this approach, the time of the utterance is known, the delay of the recognizer is known as well as the time between the utterances.

The string of instructions in the audio processing thread is summarized as follows:

1. Speaker initializes the recognition module. Recognition module measures silence and ambient noise levels of the environment
2. Speaker utters sentence
3. Recognizer recognizes sentence and saves the phonemes and their time-stamps in a file: `SOUND0001.rec`
4. The audio thread detects new output file, `SOUND0001.rec`. The system takes the time-stamp of the file
5. Speaker (same or different person) utters a new sentence and a new output file is produced, `SOUND0002.rec`
6. The audio thread detects a new output file, `SOUND0002.rec`, takes a new time-stamp and saves it
7. The difference in the time-stamps of the files is calculated and stored for synchronization
8. The process is continued from step 2

A flow graph of the process, in Fig. 5.3, makes it easier to follow. The flow chart does not include every single process in the thread in detail but only shows the sequence of important processes. To explain briefly, the audio thread starts by initializing the buffers, arrays and timing variables to be used. The timing variables are initialized to synchronize the clocks. The arrays and buffers are cleared to store the new timing information as well as the extracted phonemes from the output recognition file. The audio thread then checks for the file name that is generated to match the file

name generated by the HTK tools (SOUNDXXXX.rec). The file name is incremented by numbers in place of the "XXXX". Therefore, everytime the thread is run, it checks for whether a new file is ready to process. As soon as a new output file is ready, a time-stamp i ($\tau_{s(i)}$) is taken and stored in an array with the file number as the index. Next the information contained within the output file is processed. The phonemes are extracted and stored in an array as well as the required timing information (i.e., the time of the utterance, $\tau_{u(i)}$). With the time-stamps, the total time between the files is calculated as well as the time of the utterance. The silence time is removed from the time calculation as it was explained that the SIL time is the additional delay time for the recognizer as well as any noise the recognizer picked up. The final time used in the synchronization process is calculated using the variables in Eq. 5.4.

$$SIL = \tau_{s(i)} - \tau_{s(i-1)-\tau_{u(i)}}. \tag{5.1}$$

$$\tau_{u(i)} = \tau_{s(i)} - \tau_{s(i-1)} - SIL. \tag{5.2}$$

$$\triangle A(i) = \triangle A(i-1) + \tau_{u(i)}. \tag{5.3}$$

$$From Eq.\ 5.3: \tag{}$$

$$\triangle A(i) = \triangle A(i-1) + \tau_{s(i)} - \tau_{s(i-1)} - SIL. \tag{5.4}$$

Where $\triangle A(i)$ is the total time of the utterances minus the delay time of the processing, i.e., the amount of time needed to compensate for the audio delay in synchronization. $\triangle A(i)$ is the time the utterance was spoken and is calculated by accumulating the previous utterance times,$\triangle A(i-1)$, and adding the current duration of the utterance, $\tau_{u(i)}$. The time, $\tau_{u(i)}$, is equal to the difference in the system time-stamps, $\tau_{s(i)} - \tau_{s(i-1)}$, minus the silence, SIL which represents the delay of the recognizer for obtaining the current output (Eq. 5.3). Note that the delay of the rec-

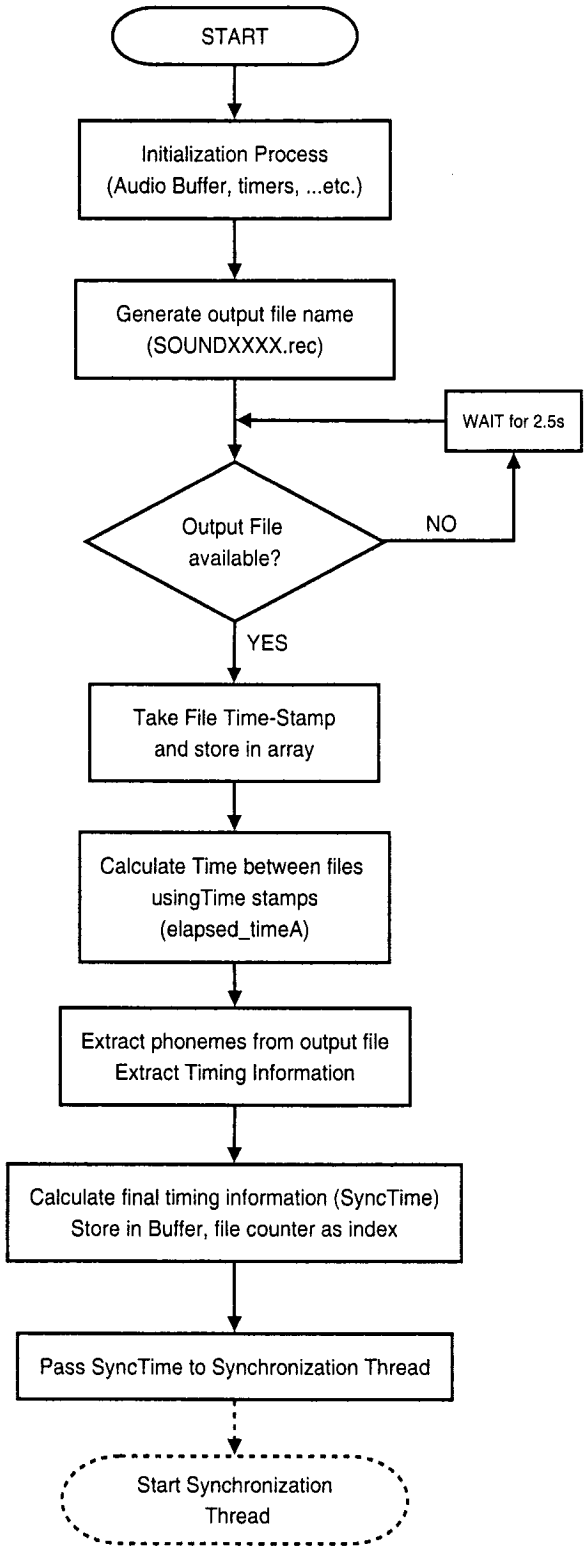


Figure 5.3: Audio Thread Flow Diagram.

ognizer, SIL, to recognize utterance i is represented by the time between the system time-stamps minus the time of the actual utterance (Eq. 5.2).

5.3 Video Processing Module

The video processing thread is responsible for capturing the raw video, applying the video processing filters, superimposing the recognized text onto the video, face localization and obtaining the timing information to calculate the video delay and in turn the frame rate that is useful in the synchronization process.

The video processing filters are applied to obtain the object edge image and also to add video processing to the system and simulate the delay associated with it. The face localization is needed to have coordinates for the text movement. Recall that the text is required to move from one speaker's face to the other speaker. The timing information is needed to evaluate the delay of the video processing and compare it to the audio processing delay and to determine the synchronization criteria which is the objective of the project.

The video is captured using a Logitech 4000 USB web camera. It is captured at a resolution of 320 by 240 pixels and using the OpenCV libraries developed at Intel for computer vision research. The video processing as mentioned in Sec. 4.3 is a series of video processing filters developed at the VidPro labs at Concordia University and the end product is the contour image. The face localization is performed using the OpenCV libraries[33].

One aspect of the system that was implemented due to the trials during implementation was the introduction of a video buffer system. From the explanation of the audio thread and the speech recognition tools in Appendix. A, it is clear that the recognized phonemes are available after the utterance is spoken and not during the speech. This means that it is impossible to synchronize the output speech with the

current video frame. For example, by the time the output recognized speech is ready to be included into the related video frame, the video frame would have been already captured and processed and is now at the frame which relates to the next speaker's speech. Therefore, a buffer system must be developed to allow the recognizer to produce an output file before the related video frame is displayed. The buffer size would be equal to the average delay of the recognizer. If the recognizer has a delay of 2 seconds then the buffer would hold 2 seconds worth of video frames before outputting the result of the video processing with the recognized text. This approach is similar to that of Benslimane[16] where he realates the buffer size to the delay between the transmitter and receiver.

A flow graph of the main processes in the video thread is illustarted in Fig. 5.4. The video thread illustarted in Fig. 5.4 starts by capturing the video frames from the camera. This is done by accessing the USB port of the machine using the OpenCV libraries[33]. The format of the capture is based on the Intel Image Processing Library (IPL) from OpenCV. The format IPL has the same structure as RGB images. The RGB images had to be converted into YUV images so that the video processing to obtain the contour image can be applied on the Y component of the image. The conversion matrix is shown in Eq. 5.5.

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}. \tag{5.5}$$

The next step is to check whether the background frame was captured. If the background frame (i.e., frame 0) was not captured then the current frame is the background and the processing of the background image is applied. The background image is needed for the remainder of the video processing (Sec. 1.2.3). The next

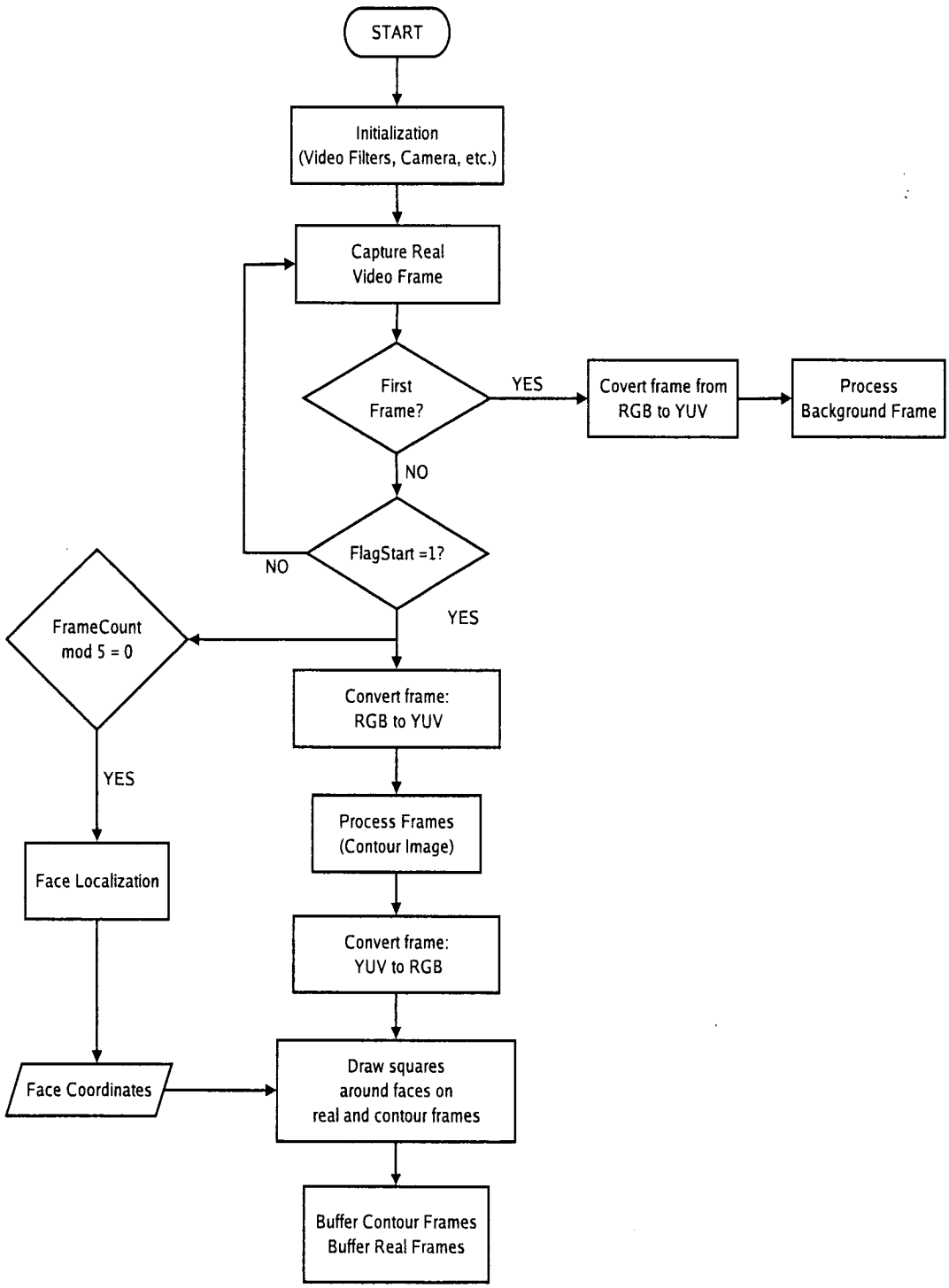


Figure 5.4: Video Thread Flow Diagram.

step is determine whether the audio thread has been initialized or not. This step is necessary to synchronize the initialization steps of both processes. The audio thread sets a flag `FlagStart` to indicate that the initialization of the recognizer is complete. The initialization of the audio thread includes noise and silence level measurements. If the flag is not set, the video thread waits until it is. After the flag is set and detected, the system continues by processing the captured frames. A frame counter monitors the number of frames processed and is stored in `FrameCount`. The face localization is performed every 5 frames as indicated in Fig. 5.4. The output of this process is the coordinates of the object's faces. Furthermore, the video thread converts the remainder of the frames to the YUV standard and continues to apply the video processing to obtain the contour image. These frames are then converted back to the RGB standard so that the OpenCV libraries can display them. The formula for conversion is given by Eq. 5.6

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1.0 & 0.0 & 1.140 \\ 1.0 & -0.394 & -0.581 \\ 1.0 & 2.028 & 0.0 \end{pmatrix} \begin{pmatrix} Y \\ U \\ V \end{pmatrix}. \tag{5.6}$$

The next step in the video thread is to draw squares around the faces using the coordinates obtained from the face localization process. All the frames, real and edge frames, are buffered to be used by the synchronization process.

Finally the thread provides the video frames that are ready to be displayed but are not synchronized with the audio thread. The only synchronization in the video thread was the initialization with the audio thread. The real frames are buffered as a reference sequence to determine objectively whether the synchronization is successful or not.

5.4 Synchronization Module

The synchronization thread is responsible for using the timing information obtained from the video and audio threads and their respective sequence of processing. As discussed in Sec. 4.3, the objective is to calculate the frame number that the text relates to according to the audio time-stamps. The video time-stamps are used to calculate the frame rate which is needed to calculate the frame number. By using the total time of the utterance, the frame number at which to insert the information is calculated by using the frame rate as explained in Eq. 4.1. Therefore, the synchronized frame, $SyncFrame$ which is the frame at which at the recognized text is to be inserted, is calculated using

$$SyncFrames = \triangle A(i) \cdot F_R. \tag{5.7}$$

$\triangle A(i)$ in Eq. 5.7 is the accumulated time of the utterances up to utterance i and is calculated using Eq. 5.4. The F_R is calculated using the video time stamps in Eq. 4.2.

The synchronization thread uses the information from Eq. 5.7 to synchronize the audio and video threads. The idea is to continuously check for the frame number and as soon as it has been detected output the relative recognized text. Since it was discussed that the video is buffered to accommodate the non-real-time characteristic of the speech recognition, each recognized utterance is also buffered along with the $SyncFrames$ associated with it. The synchronization thread controls the reading of the buffers. The synchronization thread maintains the indexing of the buffers as well as the associated output.

In the implementation there are 6 main buffers/arrays to be noted: the real frame buffer, the contour frame buffer, the phoneme (recognized text) buffer, the audio time-stamp buffer, the video-time stamp buffer, and the $SyncFrames$ buffer. All

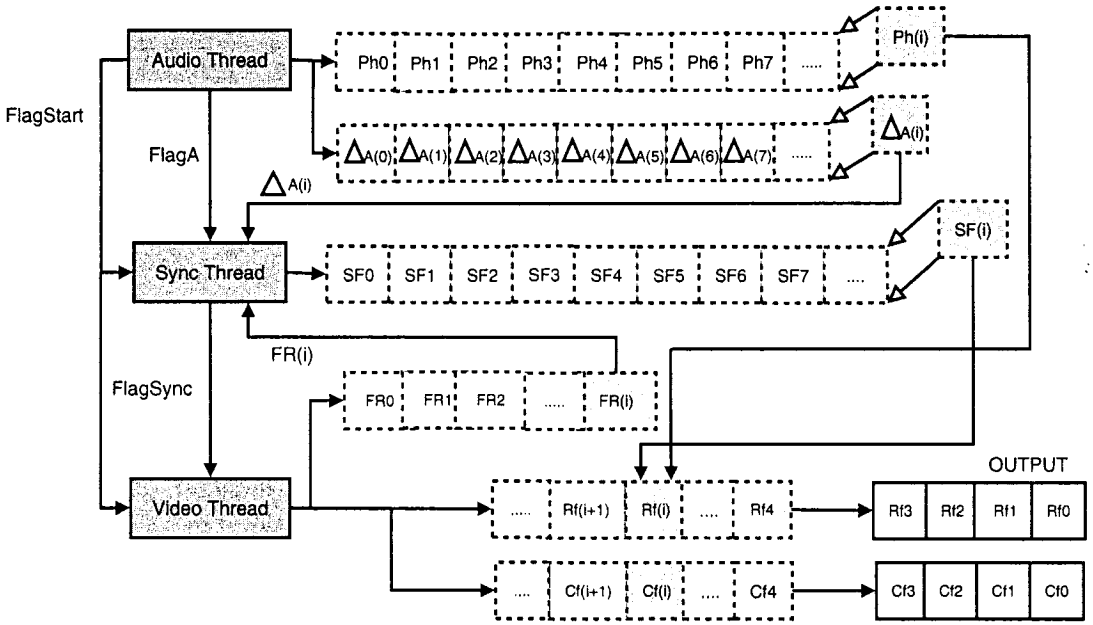


Figure 5.5: Buffer system and Communication.

the buffers are fixed in size. Since all these buffers are filled by different threads, a signalling system between the threads must be established such that incorrect access of the buffer entries are avoided. This is accomplished by using a flag system between the threads that indicate whether or not a new entry has been added and needs to be processed. The flag system integrates the different threads with each other. The flag system is illustrated in Fig. 5.5.

The audio thread is the first thread to be run as it sends the initialization signal to all the other threads *FlagStart*. This flag is set when the initialization of the recognizer is complete and the first output file is ready. The audio thread then sets *FlagStart* and the video thread starts capturing the frames and filling its respective buffer and the synchronization thread (sync thread) is ready to perform its first synchronization. The process of synchronization is started by the audio thread once again. Once the

audio thread has processed an output file, i, and extracted the timing information, $\triangle A(i)$, and the phoneme information, *Ph(i)*, another flag is triggered by the audio thread, and it is called *FlagA*.

FlagA indicates to the sync thread that a new output file is ready to be synchronized. The timing information from the file is also passed to the sync thread such that it can calculate the *SyncFrames* as indicated in Eq. 5.7. This process prompts the sync thread to read the frame rate buffer that is created by the video thread. When the new *SyncFrames*, *SF(i)* is created, a new flag is set called *FlagSync*.

FlagSync indicates to the video thread that the data for synchronization, i.e., *SF(i)* and *Ph(i)* are ready to be used. The video thread in turn reads the frame number, *SF(i)*, the frame at which to insert the text, and the text, *Ph(i)*. If the frame number has not been outputted yet, it waits until it reaches the frame number. If the frame count is larger than *SF(i)*, then video thread outputs the text immediately onto the current frame, similar to the approach in [12]. Note that the latter scenario where the frame number is larger than the *SyncFrames* indicates an error in synchronization. It is impossible to exactly catch the correct frame to superimpose the text. As soon as one de-synchronized error occurs the remainder of the synchronization is relatively incorrect because the error is accumulated. However, if the error is small then it might not be noticeable. The synchronization error has to be defined according to the application. If the error present does not affect the system, then there is no need to correct it, as explained in the work by Chen et al.[10].

However, there are approaches to correct the synchronization errors. One approach is to skip an audio output file to re-synchronize. This option is not favorable as it is obvious to the user there is a missing audio output. The second way is to repeat the last frame for the number of frames that the system is de-synchronized for. This condition is clearer when the results are explained in Sec. 6.3. This solution

is to be carried out after a certain threshold which is determined by the number of frames the audio and video are de-synchronized by. Another approach is to eliminate the accumulated synchronization error by flushing the buffers and to basically restart the system.

5.5 Summary

This chapter explained the integration methodology of the proposed system. The streams (or signals) are controlled by a thread structure. The thread structure also acts as a communication medium between the streams which governs the integration of the system. The communication allows for the synchronization thread to control the data flow in the video and audio threads and in turn have control over their synchronization. With the aid of a buffer system, the synchronization thread acts as a control center for the media streams and governs the contents of the buffer. The whole system serves as a synchronized integrated audio - video multimedia system, which is the objective.

The communication system is is set-up to work for when the audio processing delay is larger than the video delay. This is because, with the current speech recognizer, it is never going to be the case when the video processing delay is larger than the recognizer's delay. However, only a minor change in the communication and flag system is needed to accommodate the case for when the video processing delay is larger than the audio processing delay.

Chapter 6

Results

The results section discusses and displays the main results obtained first from the speech recognition and then the overall integrated system. The integrated systems results are an indication of the level of accuracy of the synchronization process. The results illustrate the effectiveness of the synchronization approach on the processed audio and video streams.

6.1 Speech recognition Results

The speech recognition results contain the accuracy of recognition from the training data, the accuracy of recognition from the voice adaptation, and the accuracy of recognition after the addition of Gaussian mixtures. The results obtained are in the form shown in Appendix A.3.

After the first output of the recognizers' training, i.e., when the 9th HMM was created, as discussed in Appendix A, the system was tested for accuracy. This test is carried out using the test directory of the TIMIT database. Testing is carried out by making the recognizer recognize the test files. The test files were not used in training the system. The HTK tool then compares the transcribed output of the recognition

to the transcription of the test files. A percentage score of recognition is obtained and is the main criteria on determining the accuracy of the recognition. The first useful output result is shown below.

```
====================== HTK Results Analysis ==============
Date:  Tue Jan 24 17:28:14 2006
Ref :  reference.mlf
Rec :  recout5.mlf
------------------------ Overall Results ----------------
SENT: %Correct=0.00 [H=0, S=1680, N=1680]
WORD: %Corr=58.58, Acc=54.90 [H=35753, D=8424, S=16851, I=2250,
N=61028]
=========================================================================
```

This shows that there was a 58.58% word accuracy in the recognition of the test data. The Acc term indicates the percentage in accuracy including the insertion errors [22]. Even though the focus of this thesis is not the accuracy of recognition, the accuracy level here was not sufficient. The recognized data must be somewhat coherent such that the delay is correctly simulated. In contrast, it is possible to create a really fast recognizor close to real-time if the accuracy score is neglected, but that will not yield any practical results. As mentioned in Appendix A, the recognizor's accuracy can be increased by modifying the number of Gaussian mixtures of the HMM (Appendix A.5).

After the incrementing the Gaussian mixtures in the process described in Appendix A.5 is performed, a new set of HMMs is obtained and the test is run again to calculate the accuracy. The results after incrementing the mixture by 2 are presented.

Gaussian Mixture	% Accuracy
5	58.58
7	61.53
9	65.35
11	67.60
13	69.16
15	69.81
17	70.35
19	70.84
21	70.96

Table 6.1: Number of Gaussian Mixture Recognition Accuracy in %

```
====================== HTK Results Analysis =======================
Date:  Tue Jan 24 17:43:12 2006
Ref :  reference.mlf
Rec :  recout7.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=0.00 [H=0, S=1680, N=1680]
WORD: %Corr=61.53, Acc=58.76 [H=37551, D=8397, S=15080, I=1693,
N=61028]
===================================================================
```

This result show an increase in accuracy by 2.95%. This shows that inceremented the number of gaussian mixtures improves recognition. The objectives here then becomes to continue incremeting the number of mixtures until the improvment in recognition between one set of HMMs and the other is negligible.

Table 6.1 states the recognition accuracy over the trials of increasing the number of Gaussian mixtures in the HMM. The last entry in the table shows a very small increase in recognition accuracy. Therefore, the trials of increasing the mixture components were halted. A 70% accuracy is sufficient for this project since the focus of this thesis is on building the whole system.

Figure 6.1 shows the graph of the results in Table 6.1. Note how the curve flattens when the number of Gaussian mixtures is above 17. This indicates that the rate of increase is very small and therefore, there is no added accuracy in increasing the

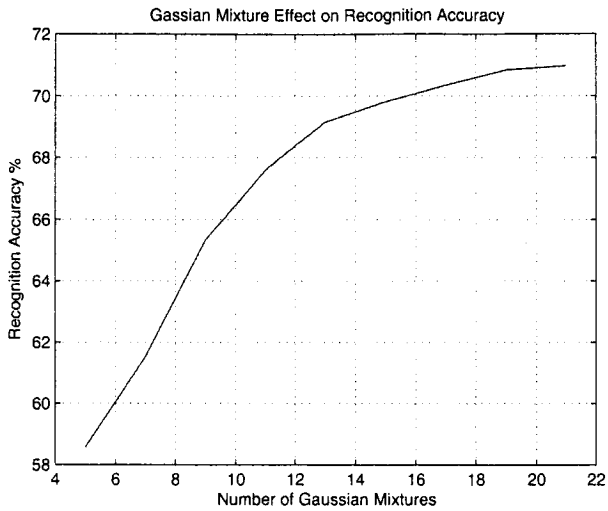


Figure 6.1: Graph of Gaussian Mixture effect on Recognition Accuracy in Percent(%).

number of mixtures beyond this point.

The next step is to adapt the HMMs to the voice of the speakers. The system was only adapted to one voice (i.e., the voice of the author) to test functionality of the recognizer. The system can be easily adapted to other voices.

The problem of the adaptation of the voice was the preparation of the sound files. The TIMIT database was prepared in speech labs where the ambient noise is minimal and the quality of recording is of industry standard. The sound files to adapt the system to the author's voice was recorded on a PC, using a standard sound card and a standard computer microphone. The adaptation process was not going to be efficient but was concrete enough for the system.

Initially, recordings of the author's were made using the HTK tool, `HSlab`. These contents of the recordings were transcribed and used in the adaptation process. Without any alteration of the sound files, the adaptation failed. The energy level, E of a signal, s is computed by

$$E = log \sum_{n=0}^{N} s_n^2. \tag{6.1}$$

Equation 6.1 is a measure of the amplitude of the signal. This energy from the speech signal is then normalized to be used in the MFCC feature extraction[31] discussed in Sec. 3.5.1. The reason the initial recordings of the author's voice failed in the adaptation process, is because they were not normalized at the same level as the rest of the TIMIT database. The energy levels in the recordings of the author's voice were much higher than that of the TIMIT database, which caused the recognizer to treat the samples from TIMIT as background noise. This meant that the TIMIT data was not used to train the recognizer and therefore, caused a really low recognition accuracy.

Therefore, the recorded speech signal energies of the author's voice were normalized the to the same approximate level in the TIMIT database. The adaptation was then successful and the testing results matched that of the last HMM in the Table 6.1.

6.2 Performance Evaluation in Related Work Papers

The performance evaluation of the speech recognition was based on the HTK tools used. The recognition accuracy used by the HTK tool, `HResult` (Appendix A), was a sufficient indication of the performance of the recognizer. This is because the focus of this thesis is the overall system and its synchronization and integration.

The performance of the system is based on how well the synchronization and integration was between the audio and video. Initially the synchronization was tested subjectively as described in Sec. 6.3. This was followed by the verification of timing parameters of the system and the errors within it. As mentioned in [10], the success

of the synchronization and integration is application dependent.

In [16], the performance of the synchronization and integration of the system is based on the timing parameters. A scenario is put forward and the system timings during synchronization and integration is compared to the scenario. The system in [16] measures the performance of synchronization under different network congestion conditions. The synchronization is evaluated between three media clients, each having their different delay times. Th final results show that the system can synchronize three different media clients with one another under moderate network congestion of 250 packets per second, up to a delay of 100ms between the clients.

In [1], the performance of the synchronization is based on the scheduled times. Tasaka et al. [1] state that there is no quantitative measure on measuring the synchronization quality[1]. The times of the synchronization are compared with predefined times. For example, the ideal output time, which is the predetermined correct time for the output of the media stream, is compared to the synchronized output time. In [1], the synchronization system run over a telecommunication network and therefore, the performance evaluators are based on the effect of the data load on the synchronization errors. For example, Fig. 6.2 show the effect of data loads on the delay of the system of different synchronization algorithms. The algorithm in , is labelled RVTR.

The work done by [1] shows the importance of measuring the media delay of the system in evaluating the performance of the synchronization. Figure 6.2 shows the delay is proportional to the data load of the network and it is at a maximum of 250ms.

In [20], the evaluation is based on the presentation time stamp (PTS) accuracy in MPEG-4 and the memory load used during synchronization. The memory load is important in [20], because the application is used to synchronize streaming video. Their results show that their adaptive mechanism provides a more accurate PTS of the media stream.

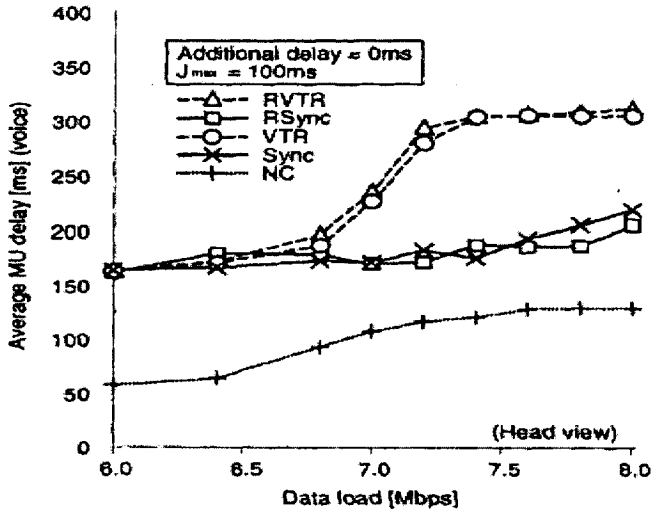


Figure 6.2: Graph from [1] showing delay times vs data load

In [12], the performance of the time-based approach for streaming video is evaluated using the timing parameters of the stream. This means that the time-stamps of capture and of playback are compared and the delay between them is evaluated. A delay threshold is set and is dependent on the experiment setup of [12]. The results compare the system's performance across an RTP network. The regular RTP network with no synchronization has a playback jitter of 246ms while the proposed approach from [12] shows a reduced playback jitter of 21ms.

The performance evaluation of synchronization systems are dependent on the timing parameters which are in turn dependent on the application. The timing parameters of the system can be defined to range within a certain threshold according the to requirements of the application. The thresholds of the timing parameters govern the degree of synchronization of the system.

6.3 Performance Evaluation: Synchronization

The synchronization of the system was tested subjectively first. This is by observing the video sequence, (real and contour), and checking whether the recognized speech was superimposed onto the frames at the time the speakers seemed to utter them in the video frame. However, this type of analysis could not be reproduced on paper.

Therefore, quantitative measures (similar to [1]) had to be produced to prove that the system achieved synchronization of the audio and video signals. There was an experimental set up to gather the information and results needed to prove that the synchronization approach was successful. The general system set up is simple: the USB camera captures the video as mentioned previously and the speakers utter predetermined sentences. The sentences are written up before hand such that the length is controlled. The length of the recognized utterance is important because the longer the sentence, the longer the recognition delay. Also note that the video buffer system depends on the delay of the recognizer, and therefore, a high delay in recognition causes the system to fail.

The objective of experimental trials are to test the systems' limit in terms of synchronization and buffer capacity. Note here that the system is tested knowing the fact that the speech recognizers delay is larger than the video processing delay. The first measurement in the trials is the time length of each utterance and the total time of speech. This is done by the aid of a stopwatch, independent from the processing of the system. The time of each utterance from the stop watch is compared to the time the recognized text that is inserted onto the frame. From the system itself, the frame rate is measured, the $SyncFrames$, from Eq. 5.7, and the difference in frames between the $SyncFrames$, (SF) and the actual frame number at which the text was inserted. This difference measure is called $DiffFrames$, (DF). Table 6.2 show the classification of the timing parameters, whether they are fixed, or dynamically

Timing Parameter	Classification
SyncFrames (SF)	Dynamic
Frame Rate (F_R)	Dynamic
DiffFrames (DF)	Dynamic
Utterance Time	Fixed
Buffer Size	Fixed

Table 6.2: Timing Parameters Classification.

measured.

There were 8 sentences used in the experiment in various order and frequency:

1. The chicken crossed the street.
2. My name is Mo.
3. The chicken crossed the road to get to the other side.
4. However, there was positive news when Wayne Rooney also ran out with the squad following his successful scan.
5. Game 3 is Saturday night in Edmonton, and the oilers will need the break to figure out some way to win a game in the series.
6. If there's a lock on the Stanley Cup, the Carolina Hurricanes hold the key.
7. Germany plays host to the world's biggest sporting event as 32 teams vie for the World Cup.
8. I got a knock on my hip and it's just made my back go into spasm a bit.

Note that the sentences vary in length such that varying recognition delay can be tested on the system. Table 6.3 shows the results of the first trial. The first 4 sentences from the list above was used in this trial. The Time column indicates the stop watch time at the end of each uttered sentence. The SF column indicates the frame number at which the text is to be inserted. The fourth column is the frame rate, FR, in frames/second. The fourth column is the incremental difference between the frame at which the text the was superimposed onto and the SF calculated from the synchronization thread, labelled DF.

The first two rows of the table show that the text was inserted at exactly the frame number, SF calculated by the synchronization thread. This indicates that there was no synchronization error. The time of the entry according to the frame rate and the

Sentence	Time(sec)	SF	F_R	DF
1	3.72	39	10.0	0
2	6.63	64	10.0	0
3	11.35	106	9.99	***-6***
4	19.25	172	9.89	***-7***

Table 6.3: Results of Trial 1

SF is $3.9sec$, which is comparable to $3.72sec$. The difference is less than 5% and could be due to human error in the timing of the sentences. The second row also shows that the there was no synchronization error in the process of inserting (and synchronizing) the second utterance. The time is also comparable, 6.4sec in playback to 6.63sec in reality.

The third row shows the trial using the third sentence. There is a synchronization error of 6 frames. This means that the text was inserted 6 frames later than calculated. This is because by the time the video thread received the SF information, the frame count outputted was greater than SF by 6 frames. At this point the system still outputs the text. The error here of 6 frames, is at a rate of 10 frames per second, producing an error 0.6sec. This is a percentage error of:

$$(DF/FR)/Time * 100 = (6/9.99)/11.35 * 100 = 5.2\%$$

This remains to be a very low error and the system is still successfull in the synchronization process. The fourth sentence produced an accumulated error of 13 frames. This number is the accumulation of the error of the previous sentence and so the real error is actually 7 frames, which converts to a time of 0.7sec, at an error rate of 7%.

The next trial was intended to test the limit of the text and video buffers and the consequence of long sentences. If the video buffer size is not adequate for the recognizers delay, the DF will increase at a higher rate, where as if the text buffer is overflowing the system will crash. Sentence number 7 was repeated and the mea-

Sentence	Time(sec)	SF	FR	DF
7	5.38	45	7.2	0
7	12.06	85	7.1	0
7	18.52	126	7.1	0
7	25.18	171	7.0	0
7	32.05	211	7.1	-6
7	38.61	254	7.1	-7
7	45.33	294	7.2	-7
7	52.13	339	7.1	-5

Table 6.4: Results of Trial 2

surements were taken. Table 6.4 displays the results.

The results from this trial show that the synchronization approach deals appropriately with long utterances however, the system runs at a lower frame rate from the processing of the large sentences. The actual synchronization is successful in most attempts and the maximum error is at(from the 6th row):

$$(DF/FR)/Time * 100 = ((7/7.1) - 38.61)/38.61 * 100 = 2.5\%$$

This error remains low even with long utterances which cause an increased delay from the recognizer. This error should be thought of in concurrence with the amount of processing the system is performing, which includes edge detection, speech recognition, and synchronization.

The next trial to be discussed evaluates the performance enhancement caused by increasing the video buffer from 20 frames to 40 frames. Recall from Ch. 4 that the buffer was implemented to compensate for the recognizer's delay. The actual size of the buffer also depends on the average delay of the recognizer. If the frame buffer is increased, then it allows the system more time to evaluate the SF and decrease the synchronization error, however, increase the overall output delay of the system. Table 6.5 compares two trials, the first with 20 frames as a video buffer, the second with 40 frame buffer.

Note that The first percentage errors were not given because the error calculated

	Buffer = 20 frames					Buffer = 40 frames				
Sentence	Time(sec)	SF	FR	**DF**	%error	Time(sec)	SF	FR	**DF**	%error
1	1.71	22	7.6	**0**	-	1.59	31	9.7	**0**	-
1	5.53	43	7.4	**0**	5.0	5.61	61	9.3	**0**	16
1	9.36	66	7.3	**0**	3.4	9.67	90	9.1	**0**	2.3
1	13.29	89	7.3	**-1**	8.3	13.76	119	9.0	**0**	3.9
1	16.86	117	7.3	**-6**	4.9	17.68	149	9.0	**0**	6.4
1	21.37	140	7.3	**-7**	10.3	21.66	175	9.0	**-1**	10.2
1	25.56	162	7.3	**-5**	13.2	25.71	201	9.0	**-9**	13.0
1	29.02	180	7.3	**-10**	15.0	29.76	230	8.9	**-6**	13.0
				-29					**-16**	

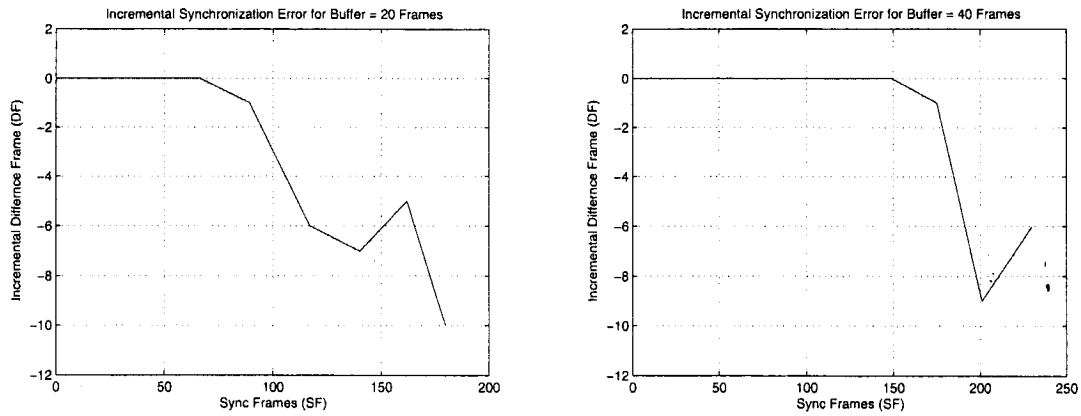
Table 6.5: Results of Buffer Trial

with the data available was not reflective of the actual events, as the time is very small and a one frame error would result in large percentage error.

Table 6.5 shows the results of the two trials involving different sized buffers, 20 frames and 40 frames. The %error is the differnece between the measured time and the calculated time from the SF and the FR. The maximum error in both cases is not over 15%. These numbers are measured in very short time periods and so are prone to large human error. The objective to test the system over short time periods, is to test the speed of the system and it's accuracy when large amounts of processing is required over a short period of time.

The trial involving the 20 frame buffer shows that the first synchronization error was present at the 4th iteration of the first sentence. The maximum error at the last iteration, was at a 15% error. In the trial involving the 40 frame buffer, the synchrnoization error was present at the 6th iteration. Therefore, it is eveident that the 40 frame buffer is less prone to error and proves that the size of the buffer affects the synchrnoization success rate. Also if the total frames of synchroniuzation error is noted (the last row in Table 6.5), notice the change in percentage error of the DF. The calculation is as follows:

20 Frame:



(a) 20 frame Buffer (b) 40 frame Buffer

Figure 6.3: Graph of Incremental Synchronization Error.

$Total\ DF = -29$

$Total\ SF = 180$

$\%error\ in\ SF = 29/(180 + 29) = 13.9\%$

40 Frame:

$Total\ DF = -16$

$Total\ SF = 230$

$\%error\ in\ SF = 16/(230 + 16) = 6.5\%$

There is a significant reduction in error caused by the increase of the buffer size, from 20 frames to 40 frames. The frame rate, FR, is also increased in the system with the larger buffer. This is due to the fact that more time is dedicated to processing rather than outputting the frames because of the buffer size. Note that in both trials the objects in the frames were identical and motion was kept at a minimum. The system as a whole performed well in this trial because of the short sentence used. Even though the succession of sentences were at close proximity in terms of time, the recognizer had a lower delay.

The graphs in Figs. 6.3(a) and 6.3(b) show that the system with a 40 buffer gives

a better performance. Both system errors fluctuates depending on the utterance and their respective time. The graph is intended to show that there is overall a lower error in the system with a larger buffer. These results can be viewed in relation with that from [1], in Fig. 6.2. Figure 6.2 showed that the larger the data, the larger the delay. The results form Figs. 6.3(a) and 6.3(b) show that a larger buffer reduces the data load on the synchronization process and in turn reduces the errors. However, these errors cannot be compared because of the differences in system characteristics.

In summary, the system has a better performance when the utterances are kept short which causes the recognizer delay to decrease. There is also a tradeoff where an increase in buffer size reduces the synchronization error, but increases the output delay.

The synchronization error is because the synchronization data, SF, is not available in time. One thing to note here is the synchronization thread depends on the audio thread and the recognizer, as explained in Fig. 5.5. If the audio thread is slow in providing the necessary flags for the synchronization operation then the probability of error in the synchronization process is higher.

6.4 Analysis of Results

The speech recognition results were satisfactory at an accuracy of 70%. This result was obtained using a large dictionary and grammer. The adaptation of the author's to the recognition was also successfull. The accuracy of the recognition was also increased by incrementing the number of gaussian mixtures in the training and testing process.

The synchronization system performed as expected and utilized the time-stamps to produce a synchronized output. The system may suffer from large delays but if the delays are controlled using an appropriate buffer size, then defects can be avoided.

$DF < 0$	Do Nothing
$DF = 0$	Insert Text
$0 < DF < (0.2 \cdot TotalFrames)$	Insert Text
$DF \geq (0.2 \cdot TotalFrames)$	Restart System

Table 6.6: Synchronization System Actions based on Difference Frame (DF)

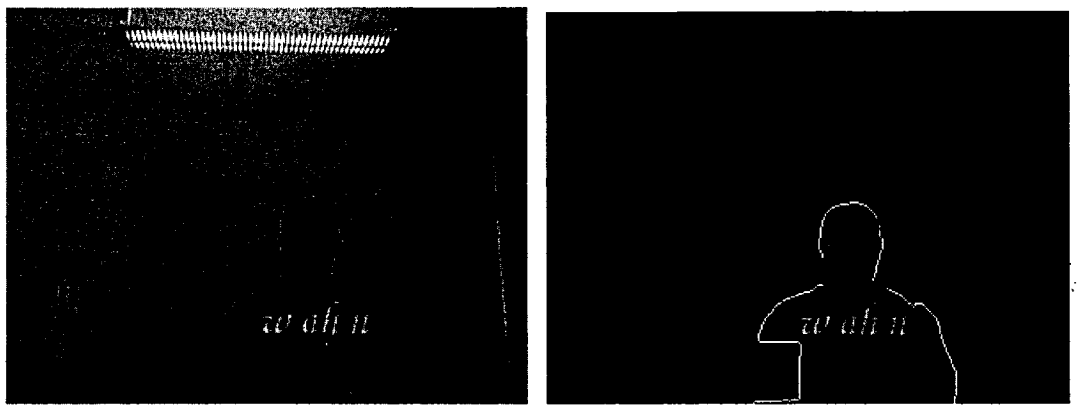
It was shown that the synchronization error was not significant. From subjective evaluation of the system output, it is determined that the system could tolerate an error of 20% because the frame rate (avg. 10 frames/second) is low. This tolerance defines the threshold Th of the system. The threshold is defined to be 20% which translates to:

$Th = 0.2 \cdot Total\ Frames$

An error above this threshold indicates the system has to be restarted or re-synchronized. Recall that this can accomplished by flushing the buffers.

Table 6.6 summarizes the actions of the synchronization system relative to the difference frames. If DF is less than zero then the system waits until the frame number to be displayed reaches the value indicated by $SyncFrames$. If DF is zero, then the text is superimposed onto the frame immediately and indicates successfull synchronization. If DF is greater than zero, but less than the synchronization error threshold Th, then the text is outputted immediately. If DF is greater than the threshold, the system has to be restarted to maintain the synchronization error to acceptable level.

If the synchronization results of the system are compared to that of Fig. 6.2, it is evident that synchronization error increases with processing load. In the case of this system, the synchronization error is increased if the utterance is long, while in [1], the delay of the media units (synchronization error) is increased with an increased data load. The overall delay differs in both systems, as there is no processing of the



(a) Real Image (b) Contour Image

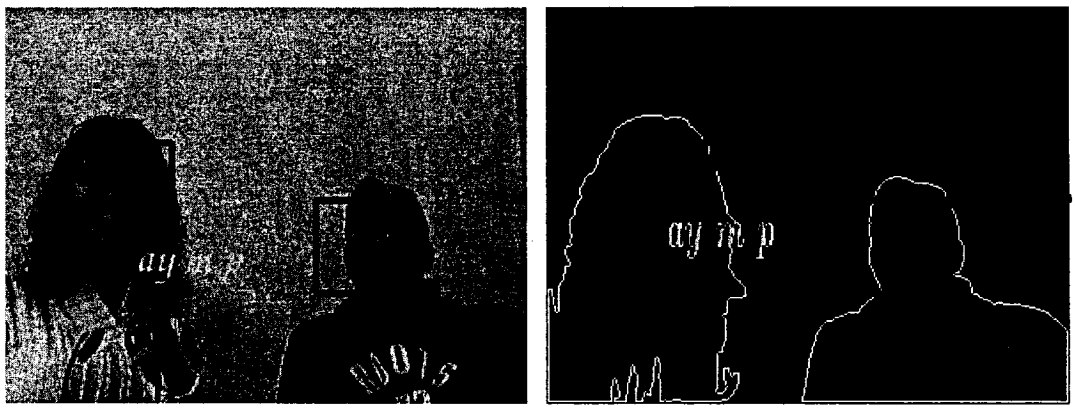
Figure 6.4: System output using One Object.

video and audio before transmission in [1], and therefore a comparison of systems is not useful.

A sample screenshot of the system output is shown in Figs. 6.4(a), 6.4(b), 6.5(a), 6.5(b), 6.6(a) and 6.6(b). The phonemes are superimposed onto the image, under the faces boxes.

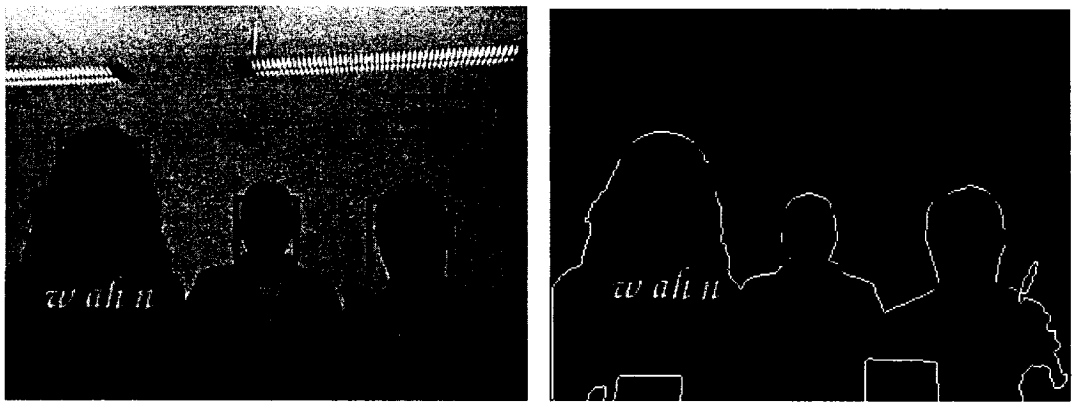
Figures 6.7(a) and 6.7(b) show the initial exchange of speech. There is only one output text from one of the speakers. The text is:"f ao r". This represents "four" in phonetics. The output text is situated next to the face box of the related speaker and travels to the next speakers face box. The Figs. 6.8(a) and 6.8(b) display the second exchange of speech. This is: " f t r iy m" which reprents "five" in phonetics. Of course the recognition is not exact and is only at 70%. The second speech text also travels to the face of the other object. The remainder of the figures show other outputs of the propsed system

The system integration was successfull and the only criteria to manage is the size of the buffers. The flag signalling set-up was sufficient to realize the systems objectives. The limitations of the system lie within the speed of the recognizer and



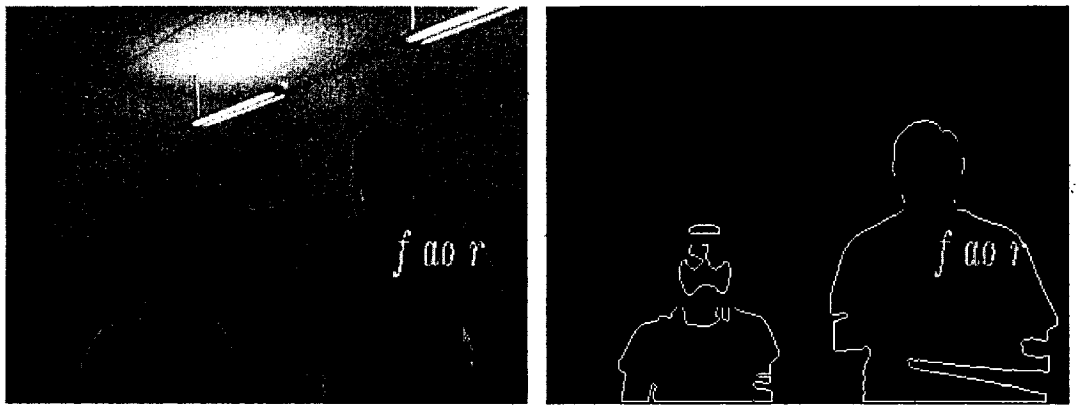
(a) Real Image (b) Contour Image

Figure 6.5: System output using Two Objects.



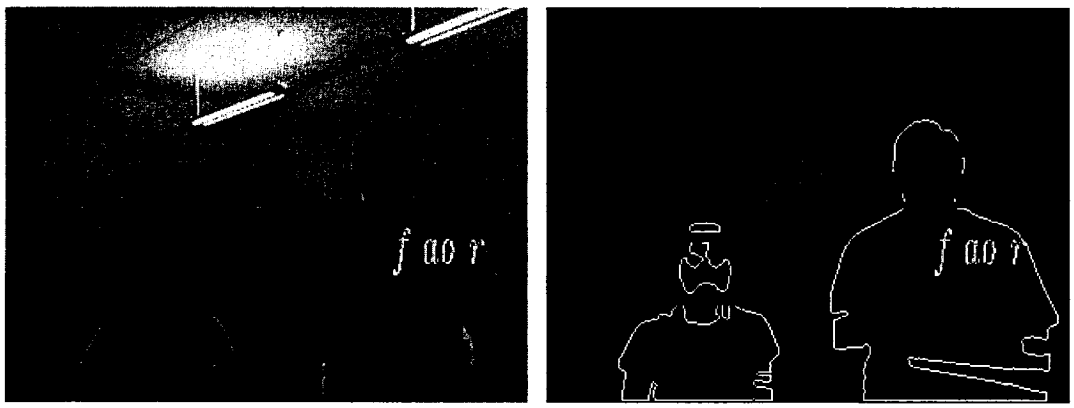
(a) Real Image (b) Contour Image

Figure 6.6: System output using Three Objects.



(a) Real Image (b) Contour Image

Figure 6.7: System output using Two Objects with First Text Output.



(a) Real Image (b) Contour Image

Figure 6.8: System output using Two Objects with Both Text Output.

(a) Real Image (b) Contour Image

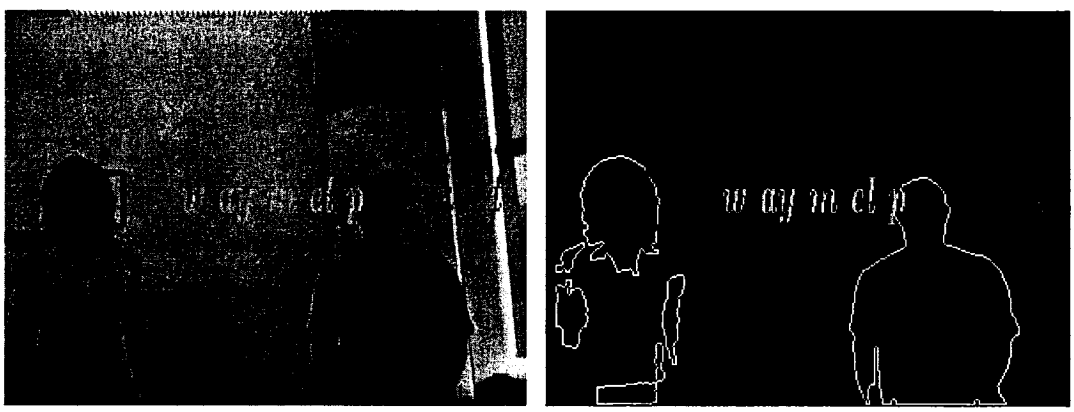
Figure 6.9: System output using Two Objects with First Text Output.

(a) Real Image (b) Contour Image

Figure 6.10: System output using Two Objects with Both Text Output.

(a) Real Image (b) Contour Image

Figure 6.11: System output using Two Objects with First Text Output.



(a) Real Image (b) Contour Image

Figure 6.12: System output using Two Objects with Both Text Output.

(a) Contour Image (first exchange of speech)

(b) Contour Image (second exchange of speech)

Figure 6.13: Contour output using Two Objects with Both Text Output.

the size of the buffers.

Furthermore, the system proposed was not dependent on the number of speakers (video objects) captured. The synchronization worked equally well for multiple speakers as it did for one speaker. The only drawback is that the speech recognizer was trained for the author's voice only, and was not adapted to other speakers. This means that the speech recognizer is speaker dependent.

Chapter 7

Conclusion

7.1 Conclusion

This thesis proposed a new approach in solving the audio - video synchronization and integration problem. The work reviewed in the Literature Review in Ch. 2 provided synchronization algorithms that used the notion of time-stamps but did not involve any processing on the raw media streams. In [21], MPEG-7 descriptors were used to solve the data retrieval problem as well synchronize the audio and video streams. Therefore, there is processing of MPEG-7 descriptors to extract the required information but there is no processing on the raw audio and video streams. Furthermore, the descriptors were dependent on the environment, and have to be designed specifically for a synchronization application. Even though it is possible to design MPEG-7 descriptors (DSs) for synchronization, it would be complicated to achieve in such a system as proposed here. This is because, the audio and video have to be separated for them to be processed, and the DSs have to be valid before and after this processing for the synchronization to be successfull. In the system proposed in this thesis, the audio stream was processed via a speech recognizer, while the video stream was processed by a video object detector. The system proposed synchronizes the audio and

video streams after processing, in any environment because the synchronization approach is dependent on the timing information. Further processing on time sensitive signals causes the synchronization of the signals to be even more challenging.

The work reviewed used the timing information of the media streams and used buffer systems to accommodate for the delay in the signals processing. Even though the work reviewed was mostly concerned with media systems that operate over a communication network, the work is still relative to what has been presented in this thesis.

The notion of using time-stamps was taken from the reviewed work ([12], [16], [21]) to establish a common reference point to the media streams. The other notion developed was that of the buffer system to store the media streams before output. The system in this thesis added the principle of processing media streams before the synchronization process and presentation. Note that several of the work reviewed use a communication system, that be an internet protocol or a control architecture to control the flow of data. These systems reviewed did not provide control systems that did not involve communication networks. The system proposed in this thesis was integrated by the use of a flag signalling system between the threads.

7.2 Summary of contributions

This thesis proposed a complex multimedia system that developed new concepts in media synchronization. The proposed system synchronized processed audio and video streams. The results show that the synchronization was successful in terms of the application. It was stated that the synchronization tolerance is application dependent and that had to be justified using objective results. It is impossible to illustrate the subjective results obtained. The results chapter (Ch. 6) provides the numbers involved in the system and displays the errors that may be present.

The system was implemented for the case of the audio processing delay being always larger than the video processing. However, because the synchronization was dependent only on the timing information, the system can adapt to both cases. Only a minor change in the integration and communication system is needed. Also it was shown (Sec. 6.4), that the system is not dependent on the number of speakers. The synchronization works equally well with multiple speakers, and only the accuracy of the speech recognition is affected, because the speakers voices were not adapted to the recognizer.

The proposed system [1] is aimed at establishing a multimedia system that uses raw media streams and produces a different output through their processing. The system achieved in developing the following:

- **Speech recognition**: developed a speech recognizer using HTK. The resources of the recognizer were developed and used to train the recognizer. A 70% recognition accuracy was established for the author's voice.
- **Audio Time-Stamps**: Extracted the timing information from the recognition. The delay of the recognizer was measured as well as the silence and utterance times.
- **Video Time-Stamps**: The delay of the video processing was measured to establish the time-stamps of the video frames.
- **Buffer**: Stored and indexed the audio data (recognized speech) and the processed video frames (contour images). The size of the buffer was dependent on delay of the particular media stream.
- **Synchronization**: Synchronization was made possible and was developed to relate the time-stamps of each stream to one another and use the buffer to output the correct data (frames and text). The general idea was to establish a common clock from the time-stamps obtained from two separate clocks (the clocks of the recognizer and the video processing).
- **Integration**: To use the synchronization framework, a signalling system was developed to be able to control the flow of data. Each of the media streams'

[1] A paper based on the proposed system was published at the Canadian Conference on Electrical and Computer Engineering 2006 (CCECE06). Two more papers are also submitted to the IEEE journal of Transactions on Multimedia and the International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007).

processing and synchronization was performed on a separate thread. A flag system was established to communicate between the threads. The flag system was used to control the flow of data and the states of each of the threads.

7.3 Possible extensions

There are a number of issues to consider in order to enhance the performance of the proposed system.

7.3.1 Speech Subsystem

The speech recognizer can be improved by adapting the system to more voices. The system will then be more robust in variations in speech. Another improvement is to speed up the recognition time as that is the bottle neck of the system. If this is accomplished, then word recognition can be implemented. Recall that word recognition (instead of phoneme recognition) was not implemented because the recognition delays were too large to be practical for the proposed system. The speech recognizer can also be improved to identify the speakers. This is by training the system and allowing the system to indicate the identity of the speaker. Again this improvement is aimed at adding more interactivity to the system.

7.3.2 Video Subsystem

As for the video processing system, it could be developed to process larger resolutions in real time. The current system is not able to process large resolutions with a small delay. The face localization can be improved to detect profile faces. At the moment the face detection is performed on a square on the face. The face localization is needed for the movement of the text.

7.3.3 Synchronization Subsystem

A number of improvements can be added to the synchronization subsystem. First, a dynamic buffer system can be implemented. The buffer sizes of the audio thread and the video thread will depend on the average processing delays of the media streams. By this the system will be even more adaptable on the implementation levels. For example, if the video delay is larger than the audio delay, then the size of the video buffer should accommodate this delay. A dynamic buffer will accommodate its size according to the average processing delays.

Then a microphone array can be used to detect the location of the speakers. The microphone array can provide the data needed to develop a Time delay of Arrival (TDOA) system that can be used to detect the location of where the speech is coming from. A microphone array system will also indicate the location of multiple speakers. By this the text can move form the speaker to the intended speaker. This would add more interaction in the system which is the objective. The microphone array will also aid in the synchronization process. The synchronization algorithm can be cross checked with the microphone array information, to locate and determine whether the speaker actually spoke or if the speech was noise. The system proposed synchronizes adequately with multiple speakers. The proposed synchronization approach is only time dependent and does not depend on the speech.

Another improvement is to correct the synchronization errors. One method is to skip an audio output file to re-synchronize the system. This option is not favorable as it will be obvious to the user there is a missing audio output. The second method is to repeat the last frame. The number of repeated frames would be equal to the number of frames that the system is desynchronized for. Another approach is to distribute the repeated frames, which will cause less obvious repetition. For example, if the error is 10 frames, then one option is to repeat every 5th frame twice. This means that the

system will be synchronized once again after the 25th outputted frame. The solution to fix the synchronization errors has to be carried out after a certain threshold which is determined by the number of frames the audio and video are desynchronized by. An example of such thresholds is given in Table 6.6. Another approach to eliminate the synchronization errors at runtime is flushing the buffers and to basically restart the system. This can be accomplished detecting a period of inactivity in the system (e.g., long periods of silence, or very little motion), and therefore flushing the buffers and restarting the system.

Furthermore, the system can be expanded to run over telecommunication networks using protocols such as RTP. Also, by implementing the proposed system using RTP, multiple sessions of the system can also be implemented. This is achieved by the synchronization algorithm proposed in this thesis to synchronize multiple media streams.

Bibliography

[1] S. Tasaka, T. Nunome, and Y. Ishibashi, "Live media synchronization quality of a retransmission-based error recovery scheme," in *IEEE International Conference on Communications, ICC*, June 2000, vol. 3, pp. 1535 – 1541.

[2] A. Amer, "Memory-based spatio-temporal real-time object segmentation," in *Proc. SPIE Int. Symposium on Electronic Imaging, Conf. on Real-Time Imaging (RTI)*, Santa Carla, USA, 2003, vol. 5012, pp. 10–21.

[3] J. Yogeshwar, "Audio/video synchronization issues," Tech. Rep., FrontPorch Digital, August 2001.

[4] Linear Acoustic Inc., "Audio and video synchronization: Defining the problem and implementing solutions, comprehensive report," Tech. Rep., Linear Acoustic Inc., 2003.

[5] Stradis Inc., "Application program interface (api) for windows," Tech. Rep., Stradis Inc., 2003.

[6] M. Robin, "The audio synchronization concept," White Paper, Miranda Inc., 1999.

[7] R. Lienhart, I. Kozintsev, and S. Wehr, "Universal synchronization scheme for distributed audio-video capture on heterogeneous computing platforms," Berkeley, CA, USA, Nov 2003, ACM Annual Conference on Multimedia, pp. 263 – 266.

[8] Advanced Communication Technologies and Services, "Vidas: Video assisted audio coding and representation," Tech. Rep., InfoWin: MPEG4 in Europe, 1999.

[9] M. Slaney and M. Covell, "Facesync a linear operator for measuring synchronization of video facial images and audio tracks," in *Neural Information Processing Systems Conference*, Denver, CO, USA., 2000, vol. 13, pp. 814–820, MIT Press.

[10] H.-Y. Chen and J.-L. Wu, "Multisync: a synchronization model for multimedia systems," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, pp. 238–248, Jan 1996.

[11] C.-M.Huang, C.Wang, and C.-H.Lin, "Interactive multimedia synchronisation in the distributed environment using the formal approach," in *IEEE Proceedings Software*, Aug 2000, vol. 147, pp. 131–143.

[12] K.H. Kim, S. Liu, M.H. Kim, , and D.H. Kim, "A global-time-based approach for high-quality real-time video streaming services," in *Seventh IEEE International Symposium on Multimedia*, 2005, p. 9pp.

[13] T. Qian, S. Tan, and R. Campbell, "An integrated architecture for open distributed multimedia computing," in *International Workshop on Multimedia Software Development, Proceedings.,*, 1996, pp. 24 – 30.

[14] Z. Splawski, "Synchronization mechanisms for multimedia streams and their specification in timed lotos," in *Proceedings of the 23rd EUROMICRO Conference: New Frontiers of Information Technology*, 1997, pp. 456 – 463.

[15] A. Boukerche, S. Hong, and T. Jacob, "A soft-handoff management scheme for wireless multimedia systems using quasi-receivers," in *International Mobility and Wireless Access Workshop, MobiWac*, October 2002, pp. 26 – 29.

[16] A. Benslimane, "A multimedia synchronization protocol for multicast groups," in *Proceedings of the 26th Euromicro Conference*, Sept 2000, vol. 1, pp. 456 – 463.

[17] K.W. Lee, D.Y. Oh, K.H. Lee, GS. Lee, T.S. Kim, and H.S. Oh, "A multimedia synchronization model for efficient service of quality," in *Proceedings of the IEEE Region 10 Conference TENCON*, Sept 1999, pp. 325 – 328.

[18] E. Stoica, H. Abdel-Wahab, and K. Maly, "Synchronization of multimedia streams in distributed environments," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, 1997, pp. 395 – 402.

[19] F. Wang, W. Zhang, and S. Yu, "Design and implementation of timing model in hdtv encoder," *IEEE Transactions on Consumer Electronics*, vol. 4, no. 4, pp. 908 – 912, Nov 2002.

[20] D. Lee, N. Kim, and S. Kim, "The mpeg-4 streaming player using adaptive decoding time stamp synchronization," in *Proceedings. Ninth International Conference on Parallel and Distributed Systems*, Dec. 2002, pp. 398 – 403.

[21] R.J. Lopes, A.T. Lindsay, and D. Hutchison, "The utility of MPEG-7 systems in audio-visual applications with multiple streams," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 1, pp. 16–25, Jan 2003.

[22] S. Young, G. Evermann, T. Hain, D. Kershaw, Gand Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book*, Cambridge University Engineering Department,Microsoft Corporation, 3.2.1 edition, 2002.

[23] B. Burchard, R. Romer, and O. Fox, "A single chip phoneme based hmm speech recognition system for consumer applications," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 3, pp. 914 – 919, Aug 2000.

[24] T. Takiguchi, S. Nakamura, and K. Shikano, "Hmm-separation-based speech recognition for a distant moving speaker," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 2, pp. 127 – 140, Feb 2001.

[25] X. Luo and F. Jelinek, "Probabilistic classification of hmm states for large vocabulary continuous speech recognition," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*. IEEE, 1999, vol. 1, pp. 353 – 356.

[26] S. Yoshiwaza, N. Wada, N. Hayasaka, and Y.Miyanaga, "Scalable architecture for word hmm-based speech recognition," in *Proceedings of the 2004 International Symposium on Circuits and Systems, ISCAS*. IEEE, May 2004, vol. 3, pp. 417 – 420.

[27] F. Jelinek, *Statistical Methods for Speech Recognition*, The MIT Press, 1997.

[28] S. Saito and K. Nakata, *Fundamentals of Speech Signal Processing*, Academic Press, 1985.

[29] M. R. Schroeder, *Computer Speech: Recognition, Compression, Synthesis*, Springer, 1999.

[30] L.R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*. February 1989, vol. 77, IEEE.

[31] C. Beccgetti and L.P. Ricotti, *Speech Recognition: Theory and C++ Implementation*, Wiley, 1999.

[32] A. Amer and C. Vazquez, "Event detection for video surveillance," Tech. Rep., vidpro-TR-03-06, Software-Copyright Report Submitted to Valeo Management L.P., Montreal, Canada, March 2006.

[33] Intel, "Open source computer vision library," http://sourceforge.net/projects/opencvlibrary/, August 2005.

Appendix A

Hidden Markov Model Toolkit (HTK)

HTK is a set of tools and applications that assist in building Hidden Markov Models (HMM). At its core, HTK can be used for the general use of HMMs. However, the extensive utilities are primarily tailored for HMM-based speech processing tools, such as continuous speech recognition applications. Other applications that can be developed using HTK include character recognition and DNA sequencing.

HTK is an open source project that is managed mainly by members of the Cambridge University Engineering Department (CUED). The Microsoft Corporation owns the license and the original code to HTK, however, Microsoft licensed the development of HTK back to CUED.

The toolkit consists of a number of library modules and tools associated with them, and are all written in the C programming language. The tools included with the kit provide processing tools for raw speech, HMM training (parameter estimation), testing and result analysis [22].

This appendix explains the main properties of the HTK toolkit. It also discuss how HTK was used to train and test the speech recognition of the system.

The HTK tools are used via the command line interface. As per any command line program, there are arguments and parameters that must be prepared in order for the program to operate correctly. In HTK. some parameters can be specified in a configuration file that can be prepared off-line. The HTK authors prefer to keep the command line interface, as opposed to a graphical one because it has the advantage of writing shell scripts which automates and facilitates the building of large-scale systems [22].

To build a system using HTK, there are four main processing steps involved. These are: data preparation, training, testing and analysis. These steps are typical for building any speech recognition system.

A.1 Data Preparation

Data needs to be prepared in order to train the system. This involves preparing sound files and their transcriptions. These files are usually obtained from database archives. HTK has a recording tool, `HSLab`, which along with recording the sound file, can be used to add the transcription.

The TIMIT database was used to train the speech recognition module discussed here. The TIMIT database is designed to provide speech data for the development of automatic speech recognition systems. TIMIT was recorded at Texas Instruments (TI), and transcribed at at the Massachusetts Institute of Technology (MIT), and prepared on CD-ROM by the National Institute of Standards and Technology (NIST). TIMIT contains a total of 6300 sentences uttered by 630 speakers from 8 major dialects of the United States.

The TIMIT database eliminates the need to record the data and their relative transcriptions. Two more steps are needed to tailor the database in the form needed for training. The first of these steps is to alter the transcriptions such that they are uniform with the grammar and dictionary to be used in the recognizer.

A.1.1 Transcription

`HLed` is a tool in HTK that is used in the data preparation. `HLed` allows for the modification of the transcriptions of the sound files. `HLed` modifies the transcriptions such that only the phones defined are present in the transcription.

All the recorded data that is used for the training process, must have the associated transcriptions defined. Even though the TIMIT database provides the transcriptions, they must be altered to accommodates the phones used. First, the transcriptions must be converted such that each word is on a separate line. All the transcriptions too are in one file, called the Master Label File (mlf).

After the mlf file is created, the phonemes are assigned to each word in the transcription. This transforms the transcriptions from a word based transcription to a phoneme based transcription. Figure A.1 summarizes the process of transforming the transcription files.

The typical command for the transformation is as follows:

```
HLEd -l '*' -d dict -i phones0.mlf mkphones0.led words.mlf
```

In the command above, `dict` contains the phoneme translation to the words used. A Typical entry in the dictionary is represented as follows:

```
ABBREVIATE ih b r iy v iy ey dx sp
```

It is basically a words-phonemes dictionary. The command looks up the transcription of the words contained in `words.mlf`, and transforms them according to the dictionary as well as any other rules which may be in `mkphones0.led`. The `-l` and `'*'` are used to generate the paths of the transcriptions. The `mkphones.led` contains the following rule:

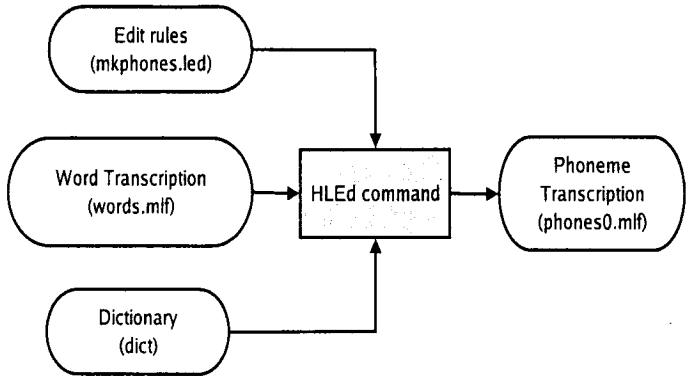


Figure A.1: Block Diagram of Transcription Process.

```
EX
IS sil sil
DE sp
```

The `EX` is a command to replace each word in the word.mlf by its corresponding phoneme translation in the dictionary `dict`. The `IS` command inserts silence (sil) models at the beginning and end of every recorded transcribed utterance. The delete command `DE`, deletes all short-pause, `sp`, labels which are not needed.

A.1.2 Feature Extraction in HTK

The `HCopy` tool , is important and must be used in preparing the raw data. Even if the sound database is obtained (as it is from TIMIT), the sound files must be converted into an appropriate format such that training can take place. Training in this sense means estimating the parameters of the HMMs according to the sound files used. Basically, `HCopy` is used to convert the source sound file into an appropriate format such that all the important information is extracted. Namely it performs feature extraction, whose theoretical process was explained in Sec. 3.5.

This stage of data preparation involves extracting the feature vectors from the raw waveforms. The method used here is Mel Frequency Cepstral Coefficients (MFCCs), which are derived from FFT-based log spectra. The `HCopy` command performs this coding. A configuration file is associated with the command which indicates which parameters are to be used. Here is the configuration file.

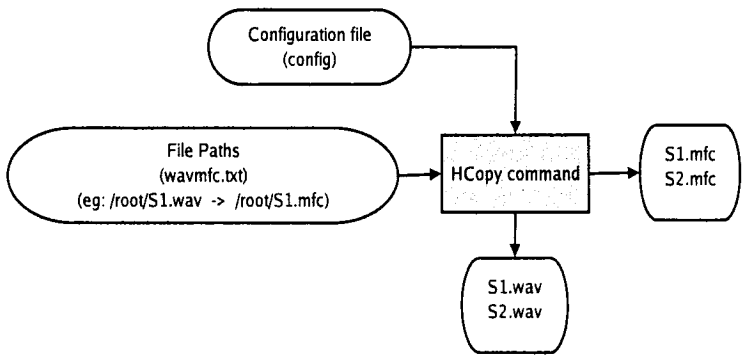


Figure A.2: Block Diagram for HCopy Process.

```
Configuration file
TARGETKIND = MFCC
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
SOURCEFORMAT = WAV
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
ENORMALISE = T
```

The `TARGETKIND` is the target coding format, and in this case is MFCC. The frame period is 10ns. The units in HTK are 100's of ns. The output is saved in a compressed format with crc checksum added. The format is in .WAV. The FFT is indicated to use a Hamming window. The filterbank is also indicated to have 26 channels and 12 MFCC filter banks. The `ENORMALISE` option indicates to the the tool to perform energy normalization on the files to be coded.

The tool `HCopy` takes in a `wavmfc.txt` file, which is a script file to indicate the path of the raw audio file and the path of the subsequent extracted feature file. An example of the command is as follows:

```
HCopy -C config -S wavmfc.txt
```

A diagram showing the structure of the Hcopy command is shown in Fig. A.2 to summarize the process[22].

A.2 Training

The training process is where the HMMs of the system are set up, and their parameters estimated according to the data provided. In HTK, HMMs are defined using text files to facilitate their editing. The initial values of the HMM are given by HTK. HTK provides a prototype for the HMM construction. The values are changed later during the training process.

A.2.1 Creating Initial HMMs

The first step is to create the prototype HMM definition. The actual values in this stage are not important but the actually topology of the HMM is defined here. The prototype is taken from the HTK manual [22], as it was the main guideline in building the HMMs. The HMM prototype used was a 3 state left to right. This is a good prototype [22] for a phone-based system, which is the objective. An example of what the prototype and the subsequent HMMs look like is given.

```
 o <VecSize> 39 <MFCC>
h "proto"
<BeginHMM>
<NumStates> 5
<State> 2
<Mean> 39
0.0 0.0 0.0 ...
<Variance> 39
1.0 1.0 1.0 ...
.
.
<TransP> 5
0.0 1.0 0.0 0.0 0.0
0.0 0.6 0.4 0.0 0.0
0.0 0.0 0.6 0.4 0.0
0.0 0.0 0.0 0.7 0.3
0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

The `"proto"` is where the phoneme is defined. In place of proto, the phonemes defined in the dictionary and the grammer is explicitly added. Between the tabs of `BeginHMM` and `EndHMM` is where the HMM of the phoneme is defined. The `NumStates` is defined as 5, including one entry and one exit state. That is why the HMM starts with defining the parameters of state 2. It defines the `<Mean>` and `<Variance>` of the state. The definition continues with remaining states. It finally defines the transition probability `<TransP>` of the states in a matrix format.

HCompV is a tool that reestimate's the global means and variances and set all the Gaussian mixtures in the HMMs to have the same values. The reestimating is based on the training data. The command is as follows:

```
HCompV -C config -f 0.01 -m -S train.txt -M hmm0 proto
```

config is the configuration file, train.txt is the script files containing the relative paths in the system to the training data, hmm0 is the folder to save the new HMM prototype proto. The -f option creates a variance floor measure along with the prototype. To create HMMs for each of the phonemes defined, the HMM prototype is copied and relabeled to the relevant phoneme, and stored in an hmmdefs file. This is a Master Macro File (mlf). By the end of this process, the HMMs for each phoneme is initialized.

The next step is to re-estimate the HMM parameters, now stored in hmm0, using the training data, obtained from the TIMIT database. The tool HRest is used to re-estimate the HMM parameters. The tool is used as follows:

```
HERest -C config -I phones0.mlf -t 250.0 150.0 1000.0 S train.txt
-H hmm0/macros -H hmm0/hmmdefs -M hmm1 monophones0
```

The new estimated values are stored in hmm1. The macros contain the global options and the variance floor values discussed earlier. This process is repeated three times, and in turn the final set of initialized monophone HMMs are present in hmm3.

A.2.2 Remodeling Silence

The previous step has generated the three state left-tot-right HMM for each phoneme and initialized their parameters. Now that the initial HMMs are defined the next step was to make sure that the HMMs are robust to noise interference. This is done by re modellling the silence representation in the HMM of sil. A state is added between the second and fourth states as shown in Fig. A.3 to represent the short pause in speech. Note this is performed in accordance with the guidelines presented in the HTK Book[22].

This is done by copying the center state of the sil model and make it the sp model. Then the tool HHEd is run to add the necessary transitions to link the sp to the sil model. The command is invoked as follows:

```
HHEd -H hmm4/macros -H hmm4/hmmdefs -M hmm5 sil.hed monophones1
```

hmm4 contains the new new set of HMMs with the sp model added. sil.hed contains the commands to tie the sp model to state 3 as indicated in Fig. A.3.

Now the HMMS are re-estimated with the new silence model. The steps discussed in the subsection of initialization are repeated after remodelling the silence. At the end of this process the HMMs are re-estimated, for a total 7 times, to be finally stored in hmm7.

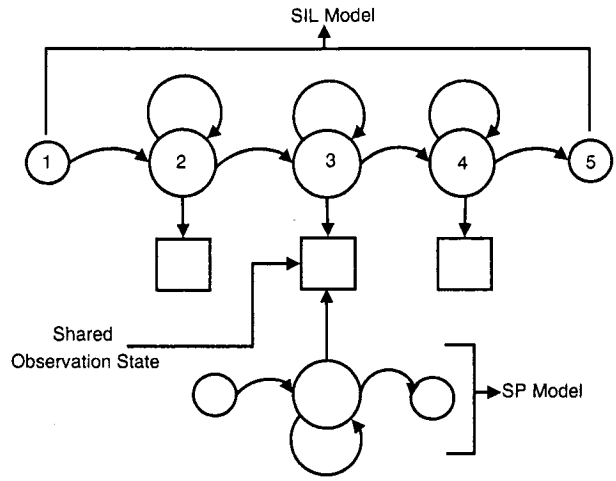


Figure A.3: Remodeling SIL HMM

Realignment

In this step, the words in the transcriptions of the training data are transformed, or realigned, to their phoneme equivalent. This is done by looking up the phonemes associated with each word in the transcription file `word.mlf` in the dictionary `dict`, and saving it in `aligned.mlf`. This is done by running the recognition, `HVite` command of HTK and is invoked as follows:

```
HVite -l '*' -o SWT -b silence -C config -a -H hmm7/macros
-H hmm7/hmmdefs -i aligned.mlf -m -t 250.0 -y lab -I words.mlf
-S train.txt dict monophones1
```

The difference between this step and the `HLEd` step discussed earlier, is that the recognizer, `HVite` considers all pronunciations of each word and outputs the most suited pronunciation according to the present acoustic training data[22].

After the realignment is completed, three more re-estimations are carried out, and the final trained HMMs would be contained in `hmm9`.

A.3 Testing

The next step in the process is the evaluation of the trained HMMs. In the TIMIT database that has been used thus far, there exists a testing set of acoustic data, as opposed to the training set. This set is used to evaluate the recognizers' performance. It contains the same sets of voices used to train the HMM. The same procedure follows for the testing set of acoustic data, where as the features must be extracted using the `HCopy` tool. To evaluate the recognizers' ability on these features the following command is invoked:

```
   HVite -H hmm9/macros -H hmm9/hmmdefs -S test.txt -l '*'
-i recout.mlf -w wdnet -p 0.0 -s 5.0 dict tiedlist
```

The HMMs used for recognition are stored in `hmm9`. The paths of the extracted features of the test data are in dictated by the script file `test.txt`. `wdnet` is the word network. The HTK recognizer requires a word network to be defined using a predefined format. The word network lists each word instance and the word to word transition [22]. It is created from the defined grammer and the HTK tool `HParse`. The `recout.mlf` file contains the output of the recognizer. The output file saves the phonemes recognized, as well as the time frames it observed them in.

Now that the output has been saved, it is compared with the original transcriptions of the test data and the score of the recognizer is obtained. The `HResults` tool is used to perform the comparison between the recognized phonemes and the transcriptions. `HResults` is invoked as follows:

```
HResults -I testref.mlf monophones1 recout.mlf
```

The results are in the form of:

```
====================== HTK Results Analysis ==============
Date:  Sun May 7 16:14:45 2006
Ref :  testrefs.mlf
Rec :  recout.mlf
------------------------ Overall Results ----------------
SENT: %Correct=98.50 [H=197, S=3, N=200]
```

The percentage value is the percentage of correct estimations between the recognized phonemes and the transcriptions. The results for the system in discussion is presented in Ch. 6.

A.4 Voice Adaptation

The system discussed up to this point has been trained from the data contained in the TIMIT database. This did not prove to be sufficient to the system. When the current HMMs trained using the TIMIT database were tested using my voice, the recognizer score was less than 20 percent. With this score the recognizer would not be able to run live, and even if it was able to produce a live output, the output delay would be very large. The large delay was unacceptable due to synchronization reasons that is explained in Ch. 4.

To increase the recognizers' ability to recognize ones voice, ideally the recognizer should be trained using that voice. However, it is unconceivable to record a set of acoustic data of ones' voice that is similar in size and quality as that of the TIMIT database. The solution is to adapt the recognizer, that is trained using the TIMIT database acoustic data, with that of sample data using the voice that is being used.

To adapt a new voice, the new acoustic data must be prepared. This is done

by generating test prompts from the dictionary being used. The training utterances must contain words that are predefined in the dictionary of the system. The tool `HEAdapt` is used to adapt the new voice and create the new HMM. The command is as follows:

```
HEAdapt -C config -g -S adapt.scp -I adaptPhones.mlf
   -H hmm16/macros -H hmm16/hmmdefs -K global.tmf tiedlist
HEAdapt -C config -S adapt.scp -I adaptPhones.mlf
   -H hmm16/macros -H hmm16/hmmdefs -J global.tmf -K rc.tmf tiedlist
```

Following the adaptation the system must again be tested. This is done using a prerecorded set of test data of the adapted voice. The testing and the results methodology is the same as explained in the previous sections.

A.5 Mixture Incrementing

Another method that was used to increase the recognizers' accuracy, was to increase the number of Gaussian mixtures in the HMM of each phoneme.

The probability distribution of the acoustic data can be modeled using a Gaussian pdf. Gaussian mixture models can be used to model the features of the acoustic data. The mixtures of pdfs is needed to model the different ways of pronouncing the phonemes, as one Gaussian pdf is insufficient.

As a method to increase the robustness of the recognizer to different pronunciations and speakers, the number of Gaussian mixtures is increased and the recognizers is re-tested and the accuracy is noted. The recognizer is re-tested to note the increase in the accuracy of recognition. This determines whether or not to continue increasing the numbers of mixtures until the required accuracy level is reached.

The tool used in HTK to increase the number of Gaussian mixtures is `HHed`. A file is created that states the rule on how many mixtures are added to the HMM. The tool is invoked as follows: `HHEd -H hmm1/MMF -M hmm2 increment.hed monophones1`

The HMM is contained in the folder `hmm9` and `increment.hed` includes the rule for added the mixture to each of the HMMs of the phonemes contained in `monophones1`.

After the mixtures have been added, the HMMs are re-estimated twice, as mentioned in the training section, then the HMMs are realigned and tested. If the results are desirable, then no more refinement of the system is needed. If the system is still not optimal to the needs, the `HHed` tool is used to add more mixtures and the process is repeated. The usual result is an increase in accuracy with the increase in mixture. However, the refinement reaches a point where adding extra mixtures results in a small increase in recognition accuracy, and further refinement will not deem usefull.