COM6906-001
160239726

The University Of Sheffield.

Individual Assessed Work Coversheet

Assessment Code: COM6906

Description: Final Dissertation

Staff Member Responsible:

Due Date: 13-09-2017 15:00:00

☑ **I certify that the attached is all my own work, except where specifically stated and confirm that I have read and understood the University's rules relating to plagiarism.**

I understand that the Department reserves the right to run spot checks on all coursework using plagiarism software.

Student Registration Number:

160239726

Assessment Code:

COM6906-001

University of Sheffield

Lip reading for song transcription

Xinghui He

Supervisor: Dr Jon Barker

A report submitted in fulfilment of the requirements
for the degree of MSc in Advanced Computer Science

in the

Department of Computer Science

September 13, 2017

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: _____

Signature: _____

Date: _____

Abstract

Lip-reading for song transcription is a specific topic on the research of visual automatic speech recognition in music. In a traditional dialogue, people recognize the speech content through their auditory sense, visual sense and also according to the circumstances with their own knowledge reserves. If compare audio recognition system to auditory sense, compare video recognition system to visual sense, compare language model to the circumstances, and compare lexicon to knowledge reserves, computer machine also can achieve the ability of recognizing the dialogue by huge training. It is a process of developing an audio-visual automatic speech recognition system.

But for this project on song transcription, it extends an MSc project that ran last year called Automatic Speech Recognition in Music[5]. This year we will focus more on the visual front end design in ASR system and compare its performance with last year's audio-only ASR system. The visual-only ASR system would be trained with traditional GMM-HMM methodologies. Since the experimental results are even more bad than last year's 91.60 percent WER with 98.49 percent WER, more information on audio-visual fusion would be covered in our paper to prepare for the future's developing on AV-ASR system in music.

Acronyms

AAM Active Appearance Model
ASM Active Shape Model
ASR Automatic Speech Recognition
AV-ASR Audio-Visual Automatic Speech Recognition
CNN Convolutional Neural Network
DCT Discrete Cosine Transform
DFT Discrete Fourier Transform
DNN Deep Neural Networks
FFT Fast Fourier Transform
GMM Gaussian Mixture Model
HMM Hidden Markov Model
LDA Linear Discriminant Analysis
LSTM Long Short-Term Memory
MFCC Mel Frequency Cepstral Coefficients
MLLT Maximum Likelihood Linear Transformation
PCA Principal Component Analysis
RNN Recurrent Neural Network
SAT Speaker Adapt Training
WER Word Error Rate

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Lip-reading is a technique of understanding speech by observing people's lip movement. And lip-reading recognition technology is a technology that integrates machine vision with natural language processing since it contributes to let machine understand language by analyzing speakers lip movement. As a simple experiment in psychology called McGurk effect[7] highlighted, auditory sense and visual sense are interacted during the process of recognizing voice. Therefore, it is reasonable to combine audio information with visual information to recognize speech. Due to the huge potentialities in applications like hearing aids or speech recognizer of silent videos and so on, researches on machine lip-reading are emerging in an endless stream nowadays. However, researches on lip-reading in music are still a barely analyzed part in visual automatic speech recognition.

For mankind, singing is an artistic way to express speech, and music is an artistic product of expressing minds. For ASR, recognition in music is more difficult since the same word has various expression in different songs even when regardless of amateur singer's accent, which may conclude different pronunciation of phoneme in our system's lexicon dictionary, CMUdict[6]. Moreover, the sound of instrument in song's background also increase the coefficients of difficulty in recognizing. Therefore, it is necessary to take advantage of visual information to recognize the speech. And in this project, we would build an visual-only ASR system in music to see whether its performance is better than audio-only ASR system. However, only use visual information would also easily get bad lip reading[1] and the method of extracting visual features affect a lot on our GMM-HMM based system's evaluation. For human, using hearing together with vision is easier than using either of them only to understand the speech.

1.1 project content

As the paper title ' lip-reading on song transcription' shows, this project would cover more researches on the visual-only ASR system in music. These can be divided as 3 main topics: audio and video feature extraction, music corpus construction and how ASR systems implemented with Kaldi[8], a popular speech recognition toolkit. In the feature extraction step, it would

refers to what feature to extract and how this feature can be extracted. And in step of music corpus construction, it would basically to state details about corpus file distribution in our system and the rules of extending the corpus, which designed by Gerardo Roa Dabike[5]. And then, this paper would also refer to some details of system experiment in Kaldi. At last, a chapter on analyzing the experimental results and conclusion on our project would be followed.

1.2 Aims and Objectives

Aiming to compare the performance of Audio-only ASR system with visual-only ASR system, this project would focus on the visual front end design to build an visual-only ASR system with the same music database last year. In our project, we would train mono-phone and tri-phone acoustic model with audio and visual feature individually and decode them with different n-gram size language model. According to the experimental results, we would see what kind of acoustic model and what size of n-gram language model would be the best choice for audio-only ASR system and visual-only ASR system respectively by comparing their WER. However, both the noise of instrument in audio the quality of ROI in video frames need to be considered as factors that affect the audio and video feature extraction individually. For they both have drawbacks in ASR of music. In the future, we will adapt the early level feature fusion to combine audio and video streams and convert them into Kaldi format in order to combine the audio with video into an AV-ASR system to see its performance.

Chapter 2

Literature Survey

Automatic Speech Recognition on music is a barely topic, since the applications on it are mostly for entertainment. Moreover, even though the research of ASR on music are small in quantity, most of them focus mainly on lyrics alignment not lyrics transcription over last decades. However, the continuous researches on music transcription come out since 2010, published by Messaros and Virtanen[2]. In terms of lip-reading, the first end-to-end model in the world based on Neural Networks called LipNet[16] comes out in 2016. It maps a variable-length sequence of video frames to text that called sentence-level lip-reading. No one can deny that it is am important breakthrough in lip-reading. And when it turns to lip-reading for song transcription, the materials that we can find are less. In order to have a basic understanding on how to build an AV-ASR system for song transcription, this Chapter would cover the literature survey on three main component : the methods of extracting audio and video features, two kinds of classification methods (GMM-HMM basically and DNN) and HMM-GMM based ASR system in music. At last, the issues on datasets and a chapter summary are also following concludes.

2.1 Feature Extraction

Feature extraction is a vital stage in our HMM-GMM based ASR systems. To extract useful feature, we avoid any meaningless features, repeated features and complicated features in principle. And try to find a suitable set with the least dimensional feature vectors to achieve the best estimation. Because low dimensional data consumes less machine computing time and also simplifies the training model. If we get a lot typical feature vectors and train a very complicated model, the training error would decrease on training set. But for the testing error, it would increase along with the rise of the complexity of training models. Therefore, how to extract feature and what kind of feature to extract has a fatal impact on system evaluation. And in this section, we will emphasize the process and methods of extracting both audio and video features.

2.1.1 Audio Feature Extraction

This section is to describe how to get the acoustic feature factors that transformed from the input waveform. And in this feature extracting process, we would particularly make research on MFCC [17](the mel frequency cepstral coefficients, a widely used feature representation in speech recognition) and use scripts in Kaldi to realize to get the feature vectors. And here is the figure 2.1 to describe the extracting process.

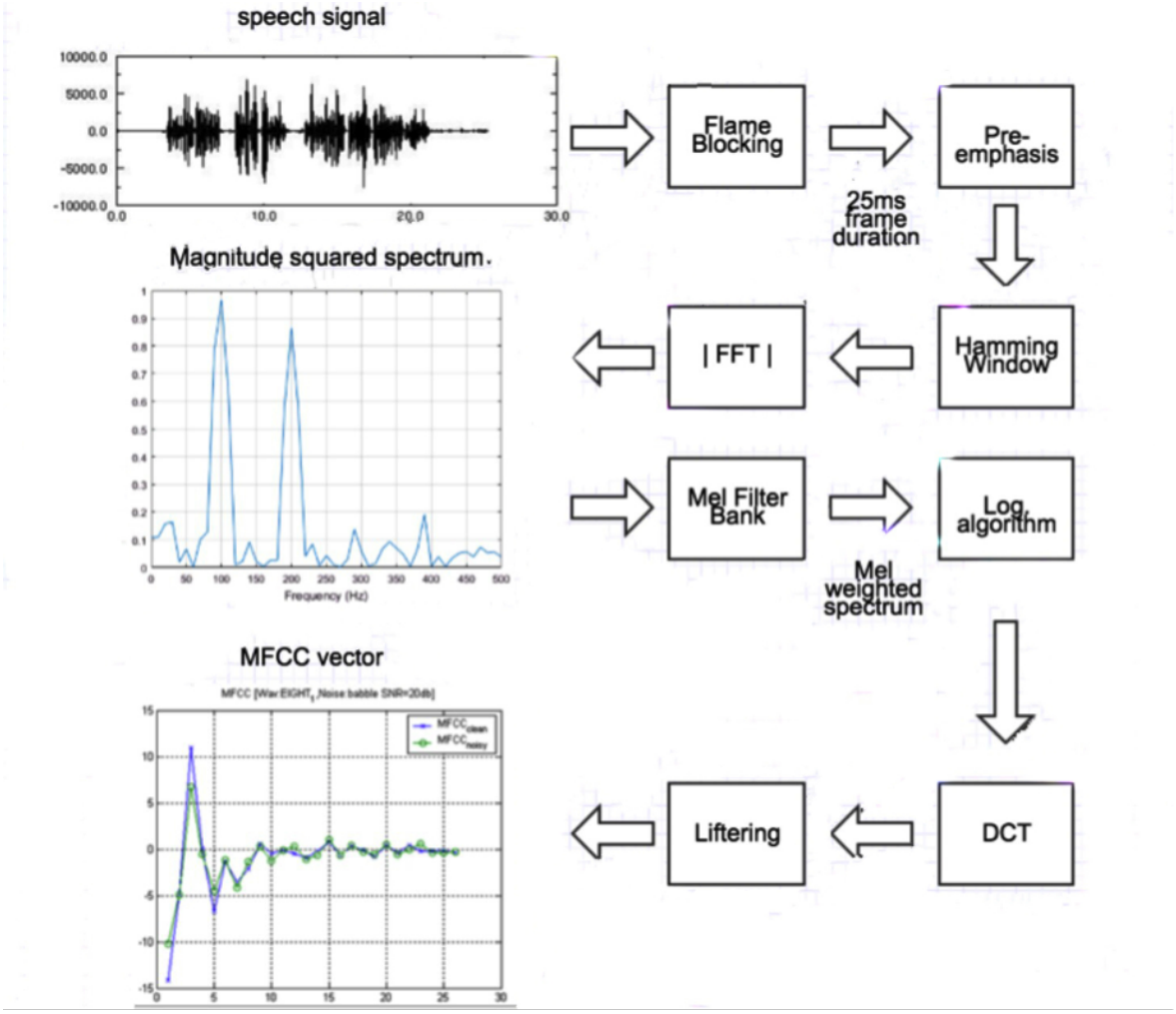


Figure 2.1: the process of extracting mfcc feature vectors

As the figure shows above, we can conclude the process of extracting mfcc feature vectors into several steps:

1. Sample and quantize the speech signal into digital waveforms.
2. Frame blocking typically 25 ms frames duration with 10ms shift.
3. Boost the waveforms energy in the part of high frequencies by using a high-pass filter.

4. Use Hamming window to make frames continuous for Fourier analysis.
5. Adapt FFT algorithm to compute the DFT in order to get the spectrum from hamming-windowed signal.
6. Collect energy from each frequency band by Mel filter bank and get fbank feature.
7. Compute the log of the spectrum values in each Mel bin.
8. Take the DCT and keep as many as MFCC coefficients.
9. Do cepstral liftering to scale the range of coefficients.

In this way, we can get standard cepstral vectors of audio. To extract dynamic mfcc features, we do delta function on mfcc to get the velocity of feature changes. Besides, if we adapt delta-delta function on mfcc, we can get accelerated velocity of feature variation .

2.1.2 Visual Feature Extraction

Visual feature can be divided as appearance-based pixel feature and shape-based feature. And in our visual-only ASR system, we use the ensemble regression tree method[14],which is implemented in dlib toolkit to extract histogram of oriented gradients feature[9] in order to detect facial 68 landmarks feature. And then we extract the according to the lip region landmarks. Here the histogram of oriented gradients feature is shape-based feature, while lip region pixel feature is appearance-based pixel feature.

Nowadays, the methods of extracting video features are mainly based on pixels or models. And this section would focus on means of models, especially on the literature of ASM and AMM, since they have been proved more informatively and visually on extracting facial features with flexible and repeated geometrical characteristic.

Active Shape Models

Active shape models (ASMs)[4] are statistical models of the shape of objects which aims to fit an example of the same object in a new image by iteratively deform. ASMs are models generally consist of two parts: A Point Distribution Model and a set of Local Gray-Level models. The former part uses objects landmark points to model the shape and its variants. Besides, the latter part is used to capture the local gray-level variants at each landmark points.

ASMs are also flexible models to represent the shape and appearance of faces in images. It uses Principal Component Analysis[12] (PCA) to learn a generic model from the facial landmarks in training data, and then fits the best match target points in test data, typically by some greedy searches.

Here are also some ASM fitting result 2.1.2 show below, which gets by initializing an approximate fit to the training data, and then updating the parameters (Xt, Yt, s, , b), applying constraints to them and repeat until convergence.



Active Appearance Model

An active appearance model [13](AAM) is a computer vision algorithm combined a statistical shape model with a texture model that aims to match a set of object shape and appearance to a new image. It is related to the ASM, since they have similar process to build models from training phrase. The difference is that the ASM seeks to match model points, while the AAM seeks to match not only the model points but also a representation of the texture of the object in an image. Therefore, the AAM can be considered as an extension of the ASM approach and be used in object detection widely nowadays. In our project, we can use tools like dlib or openCV to get the objects shape and texture information in image, and then fits the best match model in testing data, typically by some greedy searches.

Compared with ASM,our project is more reasonable to adapt AAM,since it learns model from both shape and texture information, while sometimes singer does not face to the camera. As a result, tools may be not get the landmarks. Problems get solved when adapting AMM rather than ASM.

Applying PCA, we can model the statistical variation with formula below.

A shape model:

$$X = \bar{X} + P_s b_s \tag{2.1}$$

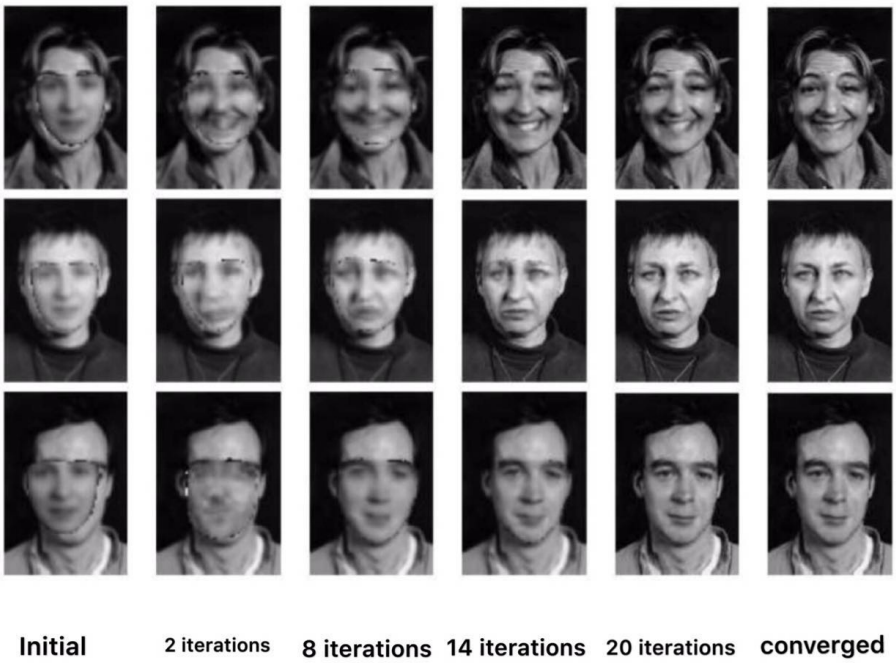
A texture model:

$$g = \bar{g} + P_g b_g \tag{2.2}$$

A shape and texture combined model:

$$b = (W)_s b_s b_g = (W)_s P_s^T (x - \bar{x}) P_g^T (g - \bar{g}) \tag{2.3}$$

Here are some examples to show the AAM fitting result2.1.2.



Audio-Visual fusion

Audiovisual fusion is a major research topic in the process of building an AV-ASR system, which aims to combine video with audio streams into a bimodal classifier. The video streams are usually starts early than audio streams, for example when the lip starts to move, the voice still not come out. Since the two streams are not perfectly synchronous with each other, the AV-fusion is complicated.

Basically, we have two methods that can be considered for AV-fusion: feature fusion, decision fusion. Feature fusion is considered as an early level fusion method, while decision fusion is a late level fusion[11]. Even though feature fusion techniques making work easier in improving ASR over audio-only result, the reliability of each modality cannot be ensured. Therefore, using a weighted decision fusion to develop the AV-ASR system may get better results than using feature level fusion. However, the decision fusion may be more difficult than feature fusion.

2.2 Classification

In this section, we would cover the literature of statistic models, HMM and GMM. They are popular models to be adapted into ASR systems. Since we did not adapt DNN methodologies into our system, the literature on DNN we covered in the chapter is rough. It is for

introduction, because using the machine deep learning methodology is the new trend to develop ASR systems.

2.2.1 Gaussian Mixture Models

The Gaussian Mixture Model [10](GMM) is a most common parametric probability density function by far, which superposed by M component Gaussian densities, to describe the distribution of the dataset. At the same time, a GMM can be expressed by equation (2.4) in a mathematical way.

$$p(x|\lambda) = \sum_{i=1}^{M} w_i g(x|\mu_i, \Sigma_i) \tag{2.4}$$

where,

$$\sum_{i=1}^{M} w_i = 1 \tag{2.5}$$

(x is a D-dimensional vector data, wi , i = 1, . . . , M, are the weights for every single Gaussian function) The shape of Gaussians controlled by the mean and unit variance, Gaussians have the same shape, with the location controlled by the mean, and here is the figure 2.2 to show the shape of one-dimensional Gaussians with different mean and variance.

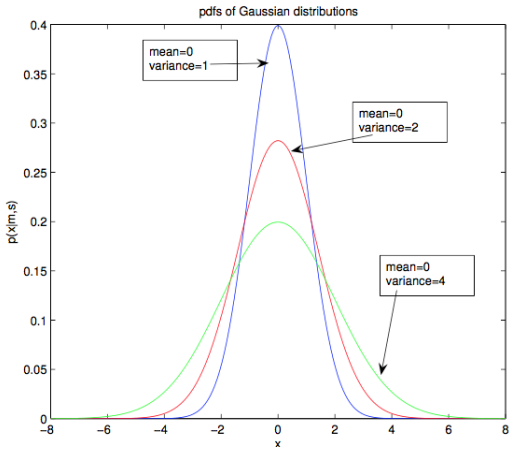


Figure 2.2: pdfs of Gaussian distributions[7]

2.2.2 Hidden Markov Model

The Hidden Markov Model [3](HMM) is a popular statistical tool in process of speech recognition, since it can model a wide range of time series data and work on both audio and video frames. We use HMM in this project, which aims to recognize the extracted audio and video features as a series of HMM state.

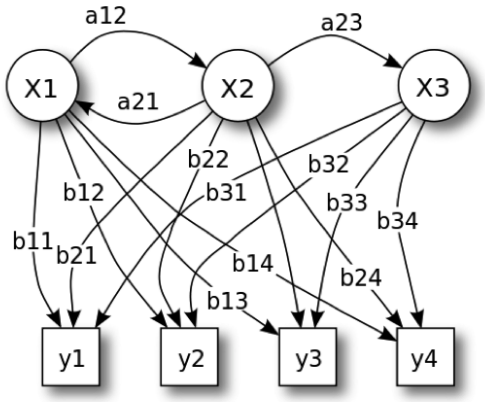


Figure 2.3: Probabilistic parameters of a hidden Markov model

As the figure 2.3 shows above, a HMM is composed by a sequence of states (x) that generates a sequence of emissions (y). And for states, there is a GMM probability density function (b) associated with each of them and also a transition probability (a) between them. In addition, we can apply some basic training algorithms in HMM for maximum likelihood estimation to get the emissions. For Viterbi Algorithm[15], it can be used to find the most likely path of getting the emissions without problems of hidden states. And for Forward Algorithm, we use it to compute the probability of the observation sequence after a given HMM. Besides, a combined version, forward-backward algorithm (Baum-Welch algorithm), can be adapted to estimate a most suitable HMM according to observed emissions with a related set of hidden states.

2.2.3 Deep Neural Networks

The neural network is a set of algorithms designed for mimicking the way that human brain works. Different with traditional machine learning algorithms, it can automatically extract features from unstructured data like audio and video frames and then cluster and classify them. Any particular Neural Networks like RNN or CNN are superposed by multiply layers while each layer composed by neural unit. Applied with an activation function, here is a figure2.4 shows how a neural unit, which from an input layer xi and weighted by a wi factor, that works.

Besides, when it comes to DNN, there are at least 3 layers: input layer, hidden layer and output layer. Moreover, every layer learns on the basic of the former layer. Therefore, the future recognized by neural unit becomes complicated along with the increasing number of layers. Here is an example 2.5 to show the process of successive model layers learn deeper intermediate representations.

And to our project, the temporal aspect is important to both audio and video component. As a result, we feed the representation of each audio or video frames to a RNN in order to

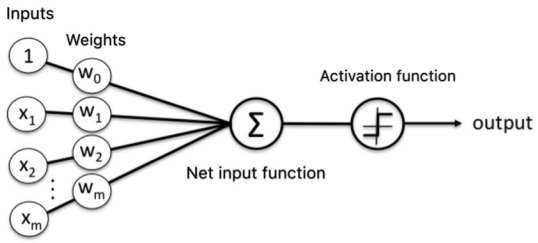


Figure 2.4: neural unit work in neural networks

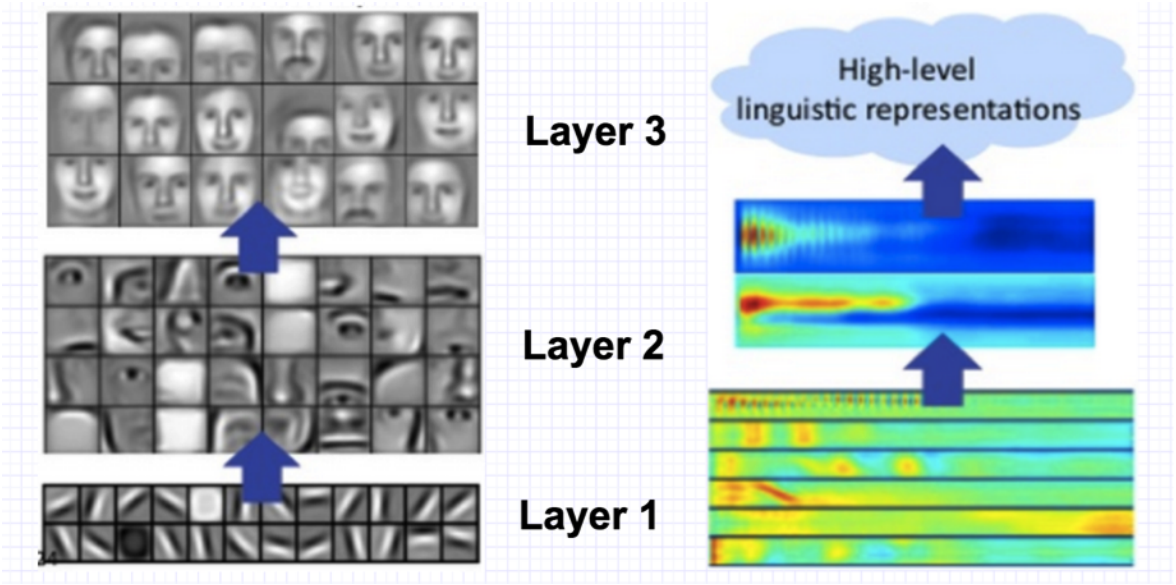


Figure 2.5: the process of successive model layers learn deeper intermediate representations

understand the sequential information between them, since a RNN is designed for recognizing sequences. And especially use LSTM to propagate and learn to control the informative streams over more time steps with forget gates. What is more, the CNN can be used to describe any individual frame and extract feature from them.

2.3 HMM-GMM based ASR system in music

Automatic Speech Recognition is a task to let machine learn how to convert the input signal into corresponded text. And in this project, we adapt a traditional HMM-GMM statistic model into our systems. As the picture 2.6 shows, HMM is to describe the dynamism with short-term stationary of states, and GMM describes the feature distribution in every state. HMM-GMM based ASR systems also have mathematic framework of them stated as below:

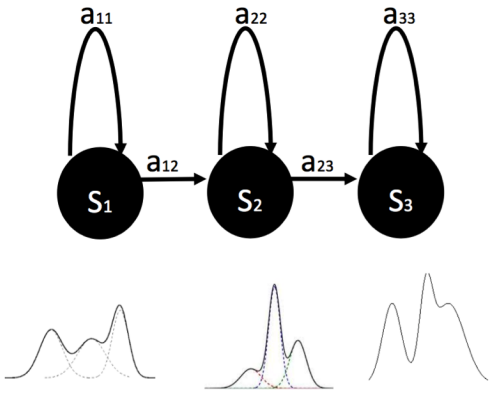


Figure 2.6: states in GMM-HMM

$$W^* = argmax P(W|X) = argmax \frac{P(X|W)P(W)}{P(X)} = argmax P(X|W)P(W) \tag{2.6}$$

W represents transcription and X represents the input signal. For X is the input signal, it is hard to calculate the formula in the first line. Therefore, we adapt bayes formula in the second line. And for the input signals are continuous, the equation can be transformed into the third line. At last, we need to find the most likely transcription W*. The HMM mathematic framework describes the process of an ASR system, w* represents the decode result, P (X—W) represents acoustic model and P (W) represents the language model. Moreover, words can be divided into sequences of HMM states Q.

2.3.1 Acoustic model

Refers to the equation above,the task of acoustic model is to compute P(X—W). No matter in audio ASR system or video ASR system, we both need to adapt their features to train acoustic models. In this project, we would train mono phone acoustic model and tri-phone acoustic model and compare their performances.

As the figures 2.7 show below, the process of training acoustic models is to model the phones with HMMs, and the states of phones are presented in normal distribution. The system would update the parameters of HMM and GMM by iterated training.

2.3.2 Language model

The language model represents the prior probability of the word sequence P(W) and is used for disambiguating between similar acoustics. For example, I want to see/sea . Sea has the same pronunciation with see, but I want to sea does not match the grammar rules. And in speech recognition, language model are statistical model and usually be called as n-gram language model. On average, bigram is the most popular size of language model implemented in ASR system. However, it depends by different systems database and would

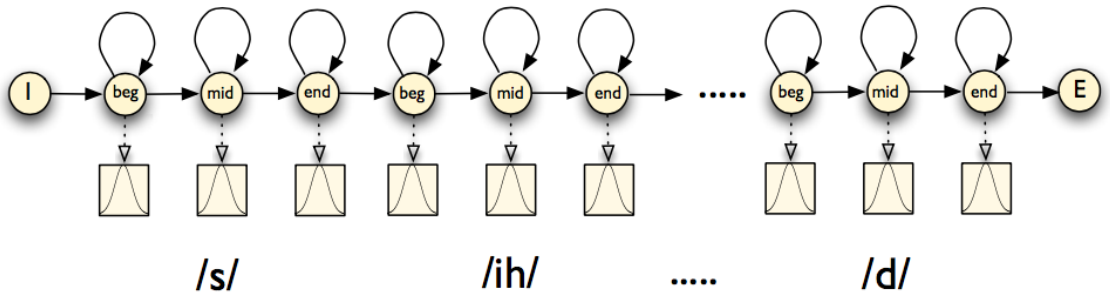


Figure 2.7: 'six quid' Word sequence models

affect the systems evaluation. We should decide the n-gram size by comparing systems result.

2.3.3 Decode ASR systems

In HMM-GMM based ASR systems, the developing process is mainly divided into 3 steps: feature extraction, classifier training, and decoding. Since the former two topics already been covered in former sections, this section is going to refer to the topic of decoding our ASR systems. To show it clearly, here is the figure 2.8 and 2.9 to show the systems construction.

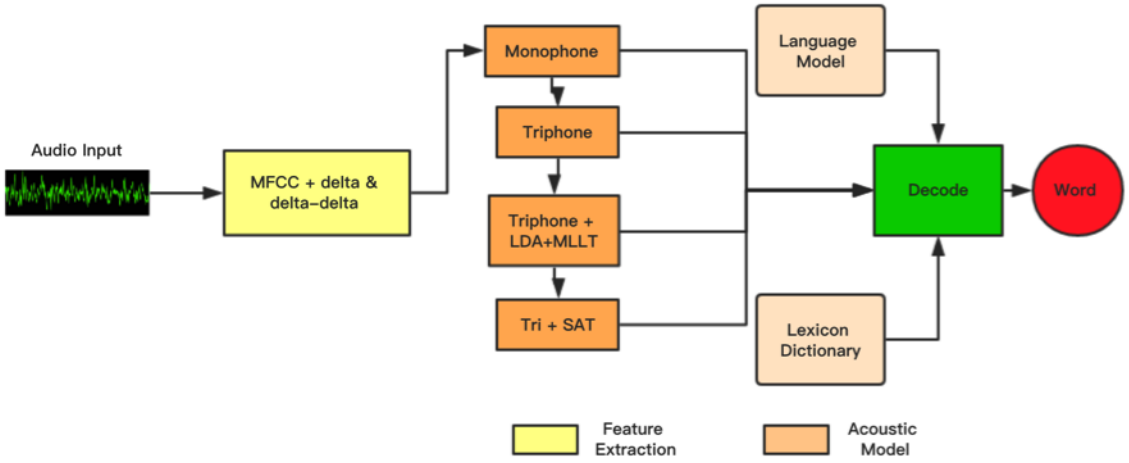


Figure 2.8: audio-only ASR system construction

In our system which is implemented in Kaldi, it would use OpenFst library to let all the resources of language model, acoustic model and grammar files are presented into fst format and generate decoding graph from them. At last, use viterbi algorithm to search the most likely path.

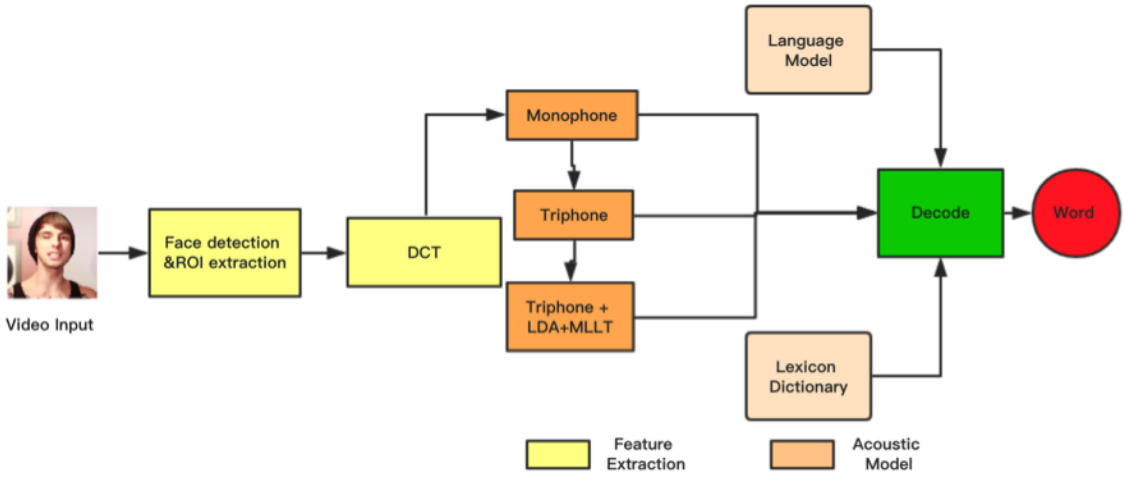


Figure 2.9: visual-only ASR system construction

2.3.4 Dataset

Speech Database VS Singing Database

Speech database is a database that concludes speech audio files and text transcription, while singing database composed by singing audio files and lyrics transcription. The singing voice is an artistic expression of speech voice spoken by singer, carrying with more information on musicality. While the speech voice tends to focus on completing the spoken word. On the other hand, the speech voice is flexible to change the pitch and loudness. But for singing voice, the frequencies fluctuate in a significantly soft way compared with speech voice. Moreover, the format of transcription between them is obviously different in the scope of topics and grammar. Text transcription may refer to any topics with a wide range and write with complicated grammar. But for lyrics transcription, the topics of it are limited. For example, probably no one would translate a science manuscript into a singing voice, while a speech voice is possible. What is more, the lyrics transcription usually has picky grammar like poems. Thus, in terms of practicability, speech database are more suitable than singing database for no matter audio-only ASR system or AV-ASR system.

2.4 summary

To conclude this chapter, we mainly refer to the literature survey on the method of feature extraction, classification and components in ASR systems. They are all useful surveys of helping to build our ASR systems with HMM-GMM methodologies. However, to keep up the pace with the Lip-reading development nowadays, we should focus more on studying deep learning method to see whether it would improve the ASR system in music. Because no matter audio-only system or visual-only system is hard to get a good result in music with HMM-GMM methods. The first end-to-end model, LipNet[8], which is learnt from deep

learning methods, comes out last year. This breaking news can be said as a milestone event in the history of lip-reading research and also accelerate the development on these researches. However, even when the researches on lip-reading are emerging nowadays, Lip-reading for song transcription is still a barely analyzed issue. Therefore, this topic still has much space to exploit on it.

Chapter 3

ACOMUS Database

The ACOMUS Database is the database we used in our ASR system in music,which is designed by Gerardo Roa Dabike. It concludes 240 English songs covered by amateur singers that are collected from YouTube. Among these songs, 200 songs are accompanied by guitar and the left 40 songs are accompanied by piano. Each song in this database has a unique song id, which composed by information of gender, sequence number of artist and instrument. For example song id:F002_002_01_0101, F002 stands for the information that the second amateur singer in database's singer list is female, the following 002 means this song is the second song in song's list, and o1 represents guitar. To look through the unique id in database, it is clear to see there are 174 amateur singers and 213 different songs in this database.

3.1 ACOMUS2 Corpus

Acoustic Cover Music Corpus 2 is a collection of isolated audio and video segments of every sentence in the singing parts of acoustic covers from amateur artist extracted from YouTube. The corpus is in English Language and the music has no restriction of style. The audio files are 16kHz sample-rated one channel files in wav format. And video files are in MPEG-4 format.For quality, the audios are accompanied with one instrument typically guitar or piano. Besides, some of the videos have low image resolution and the quantity of useful frames is small.

3.1.1 Lyrics Annotation

Lyrics is a computer file format that synchronizes song lyrics with an audio file[10]. For this project, we can only generate all the lyrics by hand for that the official lyrics would not match with the covers. We can observe the static lyrics from the website http://lyrics.wikia.com. However, it still need to check with every utterance in covers. Because the acoustic cover is not singer's official song, amateur artist may not follow the original lyrics with every utterance and sometimes they would changes some sentences in the original lyrics. Therefore, we need to modify the original lyrics to match with the cover. Moreover, we still need to segment the

songs with individual sentences with methods of marking the time when every single sentence starts and stops and store them into json format in our database firstly. And then transfer them into kaldi format, only in that way it can be read by Kaldi and used in the last stage of ASR system when scoring the WER.

3.1.2 Audio and video files distribution

As stated in the above section, the process of annotating songs consumes time and energy. In the ACOMUS Database, there are 120 songs been annotated. Even though the accumulation time of annotated songs is already not long for training acoustic models, not all the annotated utterances can be used especially in visual-only ASR system since it is picky to pick up good utterances to extract visual features. Here is the table 3.1 to show distribution of annotated utterances being used in audio-only ASR system and visual-only ASR system respectively.

System	quantity of annotated utterances	time size of annotated utterances
A-ASR system	3428	282 min
V-ASR system	1013	83 min

Table 3.1: System distribution of used utterances

3.2 Language Model

In order to generate a suitable language model for our ASR system in music. It is better to collect the lyrics with the similar style of songs in our database. Therefore, the language model is based on a lyrics corpus composed by other songs from the same 157 original singer in database. In addition, another 39 popular singers would be appended to our singer's list. At last, the lyrics corpus is generated by 25916 songs from these 196 singers.

3.3 Acoustic Model

The pronunciation of phonemes in our acoustic model is based on CMUdict, a dictionary has 39 phonemes and 133,031 different words. Aiming to keep the model to be simple, only 5000 words are selected from the dictionary. The rules of selecting these 5000 words are to keep as many as words in annotated utterance. However, there are only 3500 words can be generated from our database (among 240 songs). To complete the construction of 5k words, the left 1500 words were generated by random.

Finally, we generated the 5k words and saved it as lexicon.txt according to the annotated utterances.

Chapter 4

System implementation in Kaldi

Kaldi is a popular, open-source, speech recognition toolkit that based on weighted finite-state transducers.This chapter is going to describe how Kaldi works to build the AV-ASR system in our music corpus.

4.1 Data preparation

In our asr-music/s5 recipe, some files need to be prepared by hand and get stored in folder asr-music/s5/data, while others can be generated by Kaldi's tools like the table 4.1 below shows.

files prepared by hand	files generated by Kaldi
utt2spk	spk2utt
spk2gender	cmvn.scp
text	feats.scp
wav.scp	

Table 4.1: File distribution in prepared data

To know more about what are these data file, here are also some tables to show the format of files prepared by hand.

utterance-id	speaker-id
F002_002_01_0101.00.001	F002_01
...	

Table 4.2: utt2spk format

Methods of files generated by Kaldi

After get these data prepared, we still need to run utils/validate_data_dir.sh to check if anything already prepared well in the data directory. Sometimes it would occur some errors,

speaker-id	gender
F002_01	f
...	

Table 4.3: spk2gender format

utterance-id	annotation
F002_002_01_0101.00.001	THIS WAS ALL YOU
...	

Table 4.4: text format

like 'utt2spk is not in sorted order when sorted first on speaker-id '. Then basically we use utils/fix_data_dir.sh to fix it.

4.2 Feature extraction

This section would cover how to extract audio and video features individually and adapt them to be read by Kaldi.

4.2.1 mfcc extraction

Kaldis waveform-reading code can create MFCC and PLP features to do the ASR. In our project we choose mfcc as the audio feature, because mel scale can connect the relationship between humans sensor rate and audio actual frequency. As we know, People is easier to distinguish the slight difference in signal with low frequency, but difficult to sense the variances in signal with high frequency. Besides, as its formula $M(f) = 1125\ln(1 + \frac{f}{700})$ shows, mel scale focus more on the variances of low frequency signal.

From the table 4.6, we can see Mel frequency narrow the scope of frequency, but mostly effect on the original frequency that higher than 1000Hz. In this way, Mel scale put a large weight on low frequency signal and a small weight on high frequency signal. It is a similar process comparing with the work of humans hearing. Therefore, mfcc is more suitable to be the audio feature in this ASR system for song transcription. And after extracting the mfcc feature, it is good to do Cepstral Mean Normalisation on them to balance the average feature value.

4.3 Video feature extraction

To develop the visual front end of ASR, video feature extraction is an important step. For Kaldi doesnt provide methods for video feature, we have to extract video features individually and convert them into kaldi format in order to be read.

This process aims to extract sequences of 2D-dct transformed lip region frames of utterances. To achieve that, there are mainly 4 steps.

utterance-id	extended-filename
F002_002_01_0101.00.001	video/mp4_segments/F/F002/F002_002_01_0101.00.001.mp4
...	

Table 4.5: wav.scp format

filename	generate tools
spk2utt	utils/utt2spk_to_spk2utt.pl
feats.scp	audio:steps/make_mfcc.sh
	video:
	1.use extract_video_features.py to get utterance-id.lip_dct.hdf5
	2.convert this hdf5 file into feats.txt with Kaldi's ark format
	3.copy-feats ark,t:feats.txt ark,scp:video.ark,feats.scp
cmvn.scp	steps/compute_cmvn_stats.sh
...	

1. Detect the face in the frames and locate them as landmarks

   Here we use open-source dlib toolkit, which is implemented with the ensemble regression tree method (Kazemi and Sullivan, 2014) to extract histogram of oriented gradients feature in order to detect 68 landmarks feature. As the picture 1 shows, this is an example for dlib toolkit to detect face landmarks whose speaker id is M020 in our database.

2. Adapt the affine transform into frames

   Speaker in this picture has a frontal face to the camera, but it still has many faces can be detected with 68 landmarks but they only have side-face to the camera. Therefore, the affine transform is needed to normalize the impact of different head poses. In order to perform the affine transform, we set a landmark template, which is calculated by

frequency	Mel frequency
300 Hz	401.25 mels
1000 Hz	997.87 mels
8000 Hz	2834.99 mels
...	

Table 4.6: transform between f with Mel(f)

average frontal face landmarks in AV Lombard Grid corpus[6]. And then we compute the transform with RANSAC fitting algorithm.

3. Extract the ROI

   The ROI is a fix-sized bounding box based on the coordinates of outer lip landmarks. Because the height and width are set by 1.2 multiply (max(x)-min(x)) and (max(y)-min(y)) of these lip landmarks. Even though the size of bounding box is fixed, theyre only fixed in the same utterance of video. Therefore, we can extract the ROI and they are in RGB pixels.

4. Perform a 2d-dct on the ROI pixels to compress the frames information and remove correlation.

   Before this, it is also necessary to rescale ROI on each frame to a fixed size of 240 by 240 and convert RGB vectors into YUV color space. Finally, we set the number of max feature count as 55, and then do the diagonal scanning on DCT matrix in order to achieve the 2 dimensional dct feature with 55 elements.

4.3.1 Video feature fusion with kaldi format

To convert video feature into Kaldis format, we need to know what format the feature data present in the Kaldi. Kaldi has its script called make_mfcc.sh can generate the mfcc feature and we can check the format with Kaldis binary file called copy-feats. Here is the mfcc feature and cmvn_mfcc feature of the former 2 frames in utterance F002_002_01_0101.00.001, as we can see, every frame has 13 vectors.

```
copy-feats ark:raw_mfcc_train.1.ark ark,t:-
F002_002_01_0101.00.001  [
  76.97228 26.6251 -3.258627 -0.3438644 1.572585 4.266773 3.983776 -13.15085 -11
.18 -4.999026 -23.74091 -28.4246 1.669673
  76.31971 27.99433 -3.258627 0.6201029 0.08644867 8.415836 1.304571 -11.7375 -1
4.247 -2.416817 -26.09589 -29.15866 2.329654
```

The cmvn_mfcc feature is generated by running the script in kaldi called compute_cmvn_stats.sh in order to make mfcc features robust to some linear filtering of the signal.

```
copy-feats ark:cmvn_train.ark ark,t:-
F002_01  [
  3180926 77576.83 -230633.9 139541.5 -543193.2 33159.96 -256905.2 -273616.2 -12
1963.8 -254410.9 -328385.3 -332834 -97172.08 32547
  3.128048e+08 2426132 9846852 6312848 1.678883e+07 7525912 7935995 9104208 7600
350 7970409 9983441 9419002 5313882 0 ]
```

The next step is to transfer video features vectors with mfcc Kaldis format. The images below show the video feature vectors of the same utterance F002_002_01_0101.00.001.

```
copy-feats ark:video_train.1.ark ark,t:-
F002_002_01_0101.00.001  [
  40214.87 3962.714 -2507.24 843.1381 1323.868 1211.514 -2622.244 -1640.637 1813
.283 -70.45601 -915.1255 1025.075 -543.6578 1034.166 -521.736 757.2358 -237.5348
 1619.169 -191.2444 812.7004 -855.579 526.8428 121.7516 -163.5324 894.996 -281.6
966 683.2609 -645.5194 -1048.564 -377.0396 386.4565 -690.2971 861.5549 51.79413
126.4456 -650.1301 653.6163 769.8665 -807.6607 -574.2529 -106.5669 449.6277 -454
.5641 246.9803 -369.8257 566.6542 -3.668405 217.0995 -1.648194 193.3081 334.4752
 137.676 -186.0948 -47.19478 -198.2383
  39894.98 3992.301 -3191.384 1503.352 1972.564 595.6908 -2201.285 -2043.634 225
7.818 -308.7228 -1449.156 1053.785 -981.8397 1350.991 -824.2929 227.6725 -223.56
23 1722.68 -136.1408 908.2271 -1160.972 844.6888 130.902 46.8557 1120.289 -433.4
001 522.6392 -786.331 -821.622 -527.9636 678.5021 -74.36728 756.588 -167.0701 12
2.0844 -636.7126 -364.4191 404.1817 -480.8517 -577.8518 37.50406 87.2341 -427.91
4 281.6136 -370.9905 1151.864 358.344 49.58786 255.1052 -97.07571 179.1962 -53.9
7269 -2.224491 23.41287 -141.6693
```

```
copy-feats ark:cmvn_train_video.ark ark,t:-
F002_01  [
  1.984094e+08 1.020624e+07 -1.153829e+07 5683359 7727326 1268134 1694286 -28191
04 4601641 420643.1 -1024648 1921574 -3369639 698800.1 2528082 56004.11 -689233.
8 1253268 918768 356171.6 815366.1 235386.8 699046.2 -676096.2 695256.7 -2145502
 975971.4 -399576.9 348148.8 -167668.5 382550.7 -246625.5 -33613.98 -281629.2 28
5610.8 -369767 -70640.13 333078.2 -381824.8 -474269.4 554590.7 -628073 288389 11
241.95 -455544 56106.95 -214482 190771.5 -261876.9 109660.7 -43897.1 -182247.5 3
51089.7 -63145.75 -102618.5 6097
  7.663781e+12 3.42305e+10 1.073326e+11 1.206663e+10 1.884157e+10 1.607572e+10 4
.244467e+09 5.030039e+09 9.705627e+09 8.504186e+09 2.438386e+09 2.394439e+09 6.3
17775e+09 4.983142e+09 7.201794e+09 1.910614e+09 1.22029e+09 2.820223e+09 3.3680
14e+09 2.870343e+09 3.628668e+09 1.771182e+09 1.046794e+09 2.187298e+09 2.254213
e+09 3.888921e+09 1.860007e+09 1.965256e+09 1.740221e+09 6.703107e+08 1.522694e+
09 1.470657e+09 1.72007e+09 1.8138e+09 1.086537e+09 1.154876e+09 1.458036e+09 5.
81847e+08 1.363745e+09 1.228408e+09 1.222632e+09 1.114456e+09 1.083027e+09 6.738
127e+08 7.619245e+08 1.270366e+09 4.925417e+08 1.344474e+09 1.153543e+09 8.38920
8e+08 7.826188e+08 7.284887e+08 7.782662e+08 4.407514e+08 5.858002e+08 0 ]
```

4.3.2 Generate Language model

Since Kaldi is based on WFST framework, it provide tools to make standard arpa format represented as fst. In our recipes, we need to install and build the IRSTLM toolkit and use that to build language models from raw text. In our project, we would generate uni-gram, bi-gram and tri-gram language model, and use them respectively to decode different acoustic models.

4.3.3 Train Acoustic models

Training acoustic models with HMM-GMM method is a process of updating the parameters of HMM and GMM. In this project, we would use a few of Kaldi's scripts like train_mono.sh, train_deltas.sh, and train_lda_mllt.sh to train the acoustic models. And the detailed steps of training acoustic models can be stated as follows:

1. Initialize GMM and output o.ml and tree file.
2. Compile training graphs and output fits.JOB.gz with o.mdl, tree and L.fst as input files.
3. Align the text annotation with frames with bin/align-equal-compiled binary file.
4. Do Maximum likelihood estimate on states which are based on GMM with gmmbin/gmm-est and get the parameter of GMM.
5. Iterate the process.
6. Accumulate the states based on GMM and do Maximum likelihood estimate again to get the most likely sequence of states.
7. Merge the Maximum likelihood estimate results and generate the .mdl file.
8. Increase the number of guassians. If not surpass the number of iteration, return to the step 5 to train again.
9. Generate the final.mdl file as system's acoustic model.

4.3.4 Decode

In this section, we need decode the training data with a suitable language model matching the trained acoustic model and then create the decoding graphs. For G.fst is a presentation of language model, L.fst is a presentation of lexicon dictionary, C.fst represents the context-dependent information and H.fst represents HMM, the process of decoding is to build a Transducer from the substate of context-dependent phoneme to word with viterbi algorithm.

4.3.5 Evaluation methodology

The most common way of evaluating an ASR system is to calculate the WER. To the specific AV-ASR system in music, there are also some specific methods to evaluate the systems performance. For example, considering of the characteristics of different instrument, the effect degree to singing voice should be different as well. Therefore, we can separate songs by the type of accompanied-instrument and group them into different folders to exploit the effect of different instrument. Similar methods like pitch, tempo modification with dataset, also can be effect the audio-only ASR system's performance. For videos, we also can group songs into different folders according to the quality of video frames. Sometimes, the Lip region is partly sheltered from microphone or the frame has no face to be detected, any of them can reduce the performance of results.

On the other hand, not in terms of modifying the input data. We can update the system's WER by adjusting the phone type of acoustic model and the size of n-gram language models.

Chapter 5

Experimental results

These experimental results are shown the different performances when adapting mfcc feature in Audio-only ASR system and adapting video feature in Visual-only ASR system. As the tables below,it is clear to see, even both of the WERs are over 90%, the audio-only system still has better performance than visual-only ASR system.

Model	Uni-gram	Bi-gram	Tri-gram
Mono	93.58%	91.76%	92.68%
Tri1	92.94%	91.77%	92.09%
Tri2b	94.03%	91.60%	93.02%
Tri3b	92.39%	90.22%	90.74%

Table 5.1: WER of Audio-only ASR system

Model	Uni-gram	Bi-gram	Tri-gram
Mono	98.49%	98.82%	99.06%
Tri1	99.29%	99.58%	99.67%
Tri2b	99.58%	99.48%	99.48%

Table 5.2: WER of Visual-only ASR system

Mono model is a model that using an mono-phone acoustic model to train features. While tri1 model use tri-phone acoustic model to train dynamic features with deltas + delta-delta, tri2b model use tri-phone acoustic model to train dimensional reduced features that handled by LDA and MLLT transformations. Tri3b is a Speaker Adapted Training, therefore, we don't train visual feature with this process. And to analyze these two tables respectively, we can see that the WERs are generally lower in Audio-only ASR system when they are adapted with bi-gram language model to decode. While in terms of Visual-only ASR system, we can not make any conclusion to connect the WER results with the n-gram language model. However, they are clear to show the effects of the acoustic models. No matter decode with any n-gram language model here, the WERs in the mono-phone acoustic model are usually lower than tri-phone model.

Tri-phone models are context-dependent phone models, we adapt them because in different situations, even the same phoneme has different pronunciation.But among visual features, this problem does not exist and that make sense why visual-only system performs better when training with mono phone model.

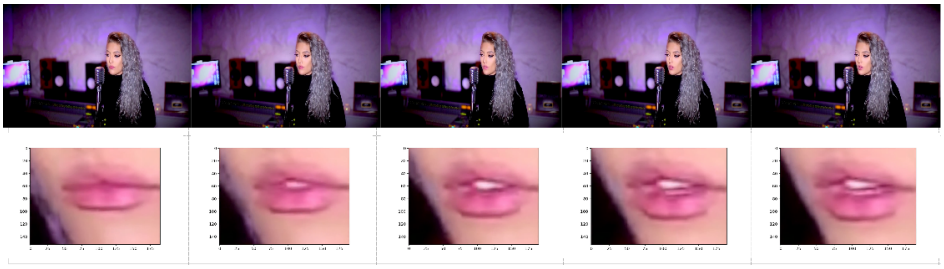
5.1 Analysis of Visual-only ASR system's result

The experimental results are too bad with over 98% WER, that is even worse than the audio-only ASR system. Audio features are not pure because it contains noise and instrument sound. In terms of visual-only ASR system, the bad result may arise from the quality of extracted features or the limited size of music corpus. To verify the assumptions, the following sections would analyze them.

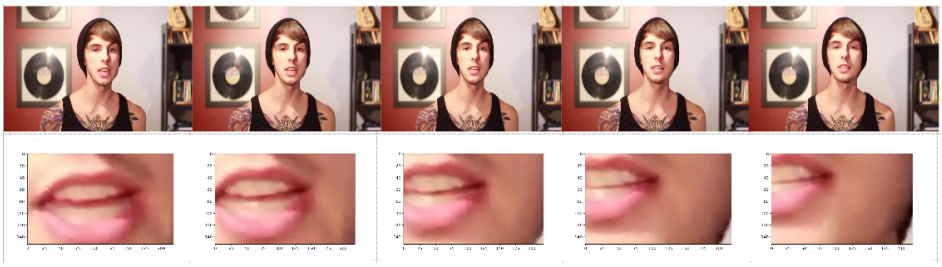
5.1.1 analysis of extracted visual pixel features

As we know, the feature extraction is an vital step in ASR systems based on statistic models. The quality of the extracted features may have heavy impacts on the decoding result. In this section, we would plot the extracted visual pixel features to see whether they are in good quality.

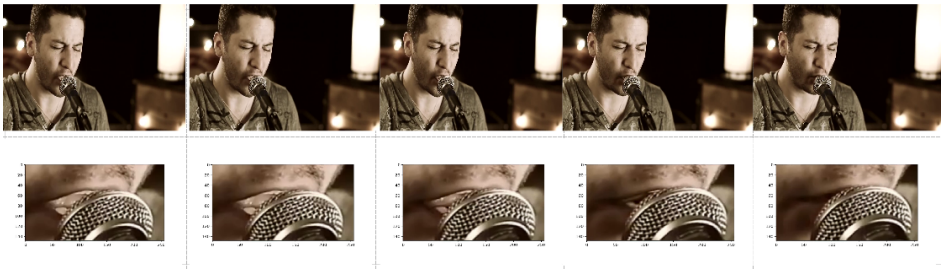
From these continuous frames in utterance: F002_002_01_0101.00.001, they are clear to show the lip_region information.



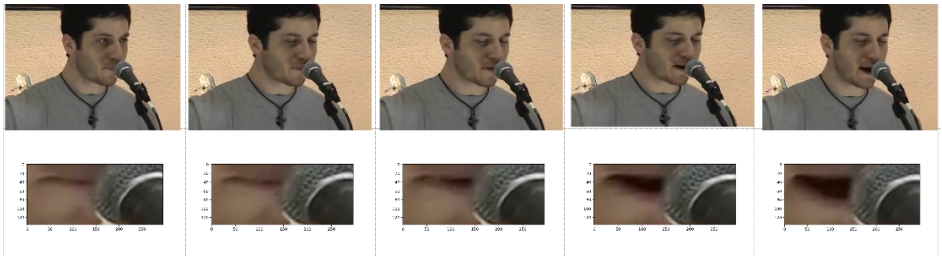
However, the most extracted features are not as good as the features above. For example, in utterance : M020_132_01_0101.00.027, the speaker moves his face quickly. As a result, the bounding box did not cover the ROI sometimes.



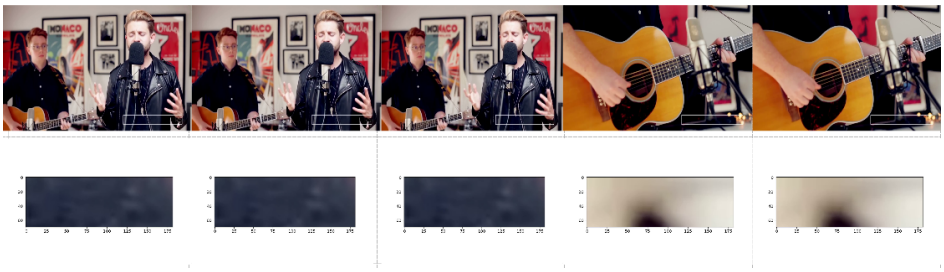
In the following example, utterance-id : M001_101_01_0101.00.022. The extracted ROI features are bad because of the shelter of microphone.



While in this utterance M001_001_01_0101.00.002 , the extracted features are more bad, since regardless of the shelter, the resolution of the video is low and may not able to show enough useful information.



Among these features, there is still another situation occurs in utterance M099_044_01_0101.00.005.



The face detector in dlib toolkit mistaken the microphone as the detected face, and extract the part of it as lip region.

The bad extracted features shown above may explain the reason why we got such bad WER results in visual-only ASR systems to some extent. To test whether the extracted features are still help in training the classifier. We take the training set data also as the testing set and get the following results. This result is indeed a improvement and it reveals that the

Model	WER
Mono	85.28%
Tri1	79.19%
Tri2b	70.38%

Table 5.3: WER of Visual-only ASR system with training and testing on the same data set

extracted visual features are still useful in training models.

5.1.2 analysis of the size of music corpus

As the table 3.1 shows, we just picked up 1013 utterances as a mini video corpus to build the visual-only ASR system. And among them only 615 utterances are for training , the left 398

utterances are taken as the testing set. Even though the WER results of Visual-only ASR system are more bad than Audio-only ASR system. We should notice that the audio-only ASR systems were trained and tested with 3428 utterances in total. Therefore, it is not reasonable to confirm which system performs better.

In the future, we should extend the useful video corpus in order to improve the Visual-only ASR system. Moreover, the algorithm of estimating lip region still can be improved, we should also focus on how to extract accurate visual features.

5.2 Bad lip reading

Bad Lip Reading is a YouTube channel,work to recompose films, TV shows, songs, sports, and political news stories by overdubbing humorous vocal that matches the lip movements of targets. It shows that, even audio-only front end system is weak in robust noise, visual-only front end system is also not reliable. Here are two videos, one is the original MV called **'youŕe beautiful'** from famous singer **James Blunt**, and the other is from this channel.

These two videos can be obtained on the website and the address would be appended in the following appendix.

5.3 Conclusion

No matter ' lip reading for song transcription' or ' audio-only ASR in music', it is difficult to achieve results with low WER. For systems those already been built, it is hard to confirm whether the audio-only ASR system performs better or not, due to the different size of training data set.And in terms of visual-only ASR system, it could have a great improvement by extending the music corpus with good quality video files in extracting features. In the future, we can also focus on combing the audio front end with the visual front end to develop an AV-ASR system to see the performance.

Bibliography

[1]

[2] Annamaria Mesaros, T. V. Automatic recognition of lyrics in singing. *EURASIP Journal on Audio, Speech, and Music Processing* (2010).

[3] Blunsom, P. Hidden markov models.

[4] Cootes, T.F., T. Active shape models-their training and application. *Computer Vision and Image Understanding* (1995).

[5] Dabike, G. R. Automatic speech recognition in music. *University of sheffield MSc project* (september 2016), 1–63.

[6] John Kominek, A. W. B. The cmu arctic speech databases. *Fifth ISCA ITRW on Speech Synthesis* (June 2004), 223–224.

[7] K. G. Munhall, P. G. Temporal constraints on the mcgurk effect. *Perception & Psychophysics 58* (January 1996), 351–362.

[8] Povey, Daniel; Ghoshal, A. The kaldi speech recognition toolkit. *IEEE Signal Processing Society* (2011).

[9] Ramanan, P. F. F. . R. B. G. . D. M. . D. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2010).

[10] Reynolds, D. Gaussian mixture models. *Encyclopedia of Biometrics* (2015).

[11] Snoek, C. G. M. Early versus late fusion in semantic video analysis. *MULTIMEDIA '05 Proceedings of the 13th annual ACM international conference on Multimedia* (2005).

[12] SvanteWold, KimEsbensen, P. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems* (1987).

[13] Taylor, T. C. . G. E. . C. Active appearance models. *Published in: IEEE Transactions on Pattern Analysis and Machine Intelligence* (2001).

[14] Vahid Kazemi, J. S. One millisecond face alignment with an ensemble of regression trees. *IEEE Xplore* (2014).

[15] Viterbi, A. A personal history of the viterbi algorithm. *IEEE Signal Processing Magazine* (2006).

[16] Yannis M. Assael, Brendan Shillingford, S. W. N. d. F. Lipnet: End-to-end sentence-level lipreading. *Department of Computer Science, University of Oxford, Oxford, UK* (2016).

[17] Zheng Fang, Zhang Guoliang, S. Z. Comparison of different implementations of mfcc. *Journal of Computer Science and Technology* (2001).

Appendices

Appendix A

Details of experimental result

Detail of the insertions, deletions and substitutions words

Uni-gram	Model	total words	Insertions	Deletions	Substitutions
	Mono	2121	14	1468	607
	Tri1	2121	3	1673	430
	Tri2b	2121	2	1817	293
Bi-gram	Model	total words	Insertions	Deletions	Substitutions
	Mono	2121	10	1467	619
	Tri1	2121	5	1494	613
	Tri2b	2121	1	1858	251
Tri-gram	Model	total words	Insertions	Deletions	Substitutions
	Mono	2121	14	1370	717
	Tri1	2121	4	1476	634
	Tri2b	2121	2	1785	323

Table A.1: Detail of the insertions, deletions and substitutions words of Visual-only ASR system

Bi-gram	Model	total words	Insertions	Deletions	Substitutions
	Mono	4008	29	2036	1353
	Tri1	4008	21	2375	778
	Tri2b	4008	1	2320	500

Table A.2: Detail of the insertions, deletions and substitutions words of Visual-only ASR system trained and tested by the same data set