ABSTRACT

Title of thesis: AUTOMATIC SPEECH CODEC IDENTIFICATION WITH APPLICATIONS TO TAMPERING DETECTION OF SPEECH RECORDINGS

Jingting Zhou, Master of Engineering, 2012

Thesis directed by: Professor Carol Espy-Wilson
Department of Electrical and Computer Engineeing

In this work we investigated source-dependent techniques for speech media authentication. In partcular, we developed algorithms to detect wheather the speech was coded and, if so, which CELP-based codec was used. Finally, we demonstrated that this knowledge can be used to determine if a speech utterance has been tampered with.

AUTOMATIC SPEECH CODE IDENTIFICATION WITH APPLICATION TO TAMPERING DETECTION OF SPEECH RECORDINGS

by

Jingting Zhou

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Engineering
2012

Advisory Committee:
Professor Carol Espy-Wilson, Chair/Advisor
Professor Shihab Shamma
Professor Min Wu

Table of Contents

List of Figures

Chapter 1

Introduction

The objective of speech media authentication (SMA) is to establish the validity of a speech recording as a true "acoustic representation" of an event that occurred at a certain time and place. Particular applications of this process are embodied in the answers to the following common questions: (i) is the recording original or a copy; (ii) has the recording been edited or modified since its creation; (iii) does it come from the alleged source; and (iv) is the content consistent with what is known or alleged. The most general framework towards SMA is the blind-passive approach. Algorithms based on this approach do not rely on the presence of a watermark or extrinsic fingerprint, but on the traces left behind by the generating process and signal modifications. Two different types of information can be targeted for SMA: (i) source dependent, where the extracted information is directly tied to the intrinsic fingerprint of the source; and (ii) source independent, where the information is not directly related to the source (i.e., background noise, electric network interference, etc). Once this information has been automatically extracted, a consistency test or anomaly detection procedure can be used to extract evidence relevant for the authentication task at hand.

For source dependent approaches, Garcia-Romero [1] explored the intrinsic variability of recording devices and use a statistical model, contextualized by speech

content, to describe them. A universal background -Gaussian mixture model(UBM-GMM) is trained and for each device, a GMM is adapted from the UBM. The means of one GMM are appended together to construct a device ID. These dev-IDs are found to be very discriminative. Scholz [2] developed an algorithm to detect a wide range of state-of-the-art speech codecs. By subtracting the harmonic structure, the noise spectrum was obtained and served as input to a support vector machine, classifier to determine which of five different codecs was used. Yang [3] examined the compression mechanism of mp3 files as a way of detecting forgery. When an mp3 file is encoded, the audio samples are divided into frames and each frame has its own frame offset after encoding. By examining the trace left by the quantization process, the frame offset can be detected with high accuracy. Forgeries will break the original frame grids, thus leave evidence of the manipulation.

For source independent approaches, Farid [4] explored the continuity of natural speech and assumes that a "natural" speech signal has weak higher-order statistical correlations in the frequency domain, and bispectral analysis would reveal "un-natural" forgery in speech. Grigoras [5] proposed electric network frequency(ENF) information can be used to reveal the timestamp of the audio. ENF refers to the magnitude of 50/60 Hz network frequency signal, captured by digital equipment when the audio is recorded. The magnitude of network frequency signal is fluctuating all the time, by comparing the fluctuation pattern with a reference pattern from the electric company, the exact time of the recording can be verified.

The focus of this work is on source dependent techniques. In particular, we are interested in performing speech media authentication following a two step process.

The first step involves detecting the type of speech codec used to generate the signal. The second step uses known properties of the detected codec to perform media authentication. Our focus will be on recordings of speech signals that have been encoded with members of the CELP family of speech codecs [6,7,8,9].

Chap. 2 gives a brief introduction on speech codecs. Chap. 3 describes the codec detection algorithm in detail. Chap. 4 gives an improved algorithm for speech files from SILK, the speech codec for Skype calls. Chap. 5 introduces a tampering detection algorithm based on a proposed codec detection algorithm.

Chapter 2

CELP Family of Speech Codecs

Spanias [10] presented a good summary on speech codecs. The Code Excited Linear Prediction(CELP) codec is the most popular one in the cellphone network. There are many versions of CELP.

Figure 2.1 shows a block diagram of the decoding process of a CELP codec. $\mathbf{v}_{fixed}$ is a vector from a fixed codebook stored in the memory of the decoder, and it captures the aperiodic portion of the residual signal, so its energy is high in unvoiced regions. $\mathbf{v}_{adap}$ is a vector copied from an adaptive codebook which contains previously reconstructed residuals, and it captures periodic portions of the speech signal so the energy is high in voiced regions. The weighted sum of these two vectors, the reconstructed residual r, is fed into the inverse LPC filter. The corresponding weights for $\mathbf{v}_{adap}$ and $\mathbf{v}_{fixed}$ are a_{adap} and a_{fixed}, respectively. The output of the post-processor is the decoded speech.

Different versions of CELP have different codebooks, i.e. different kinds of $\mathbf{v}_{fixed}$. Figure 2.2 shows some examples of $\mathbf{v}_{fixed}$ from different codebooks. As we can see, $\mathbf{v}_{fixed}$ from EFR contains 10 pulses of the same magnitude, and for $\mathbf{v}_{fixed}$ from SILK, the ratio of the peak magnitude (around 800 as in the figure) over the non-peak magnitude (around 200) is always 4. All the details of these codebooks are described in the codec standards document or the implementation

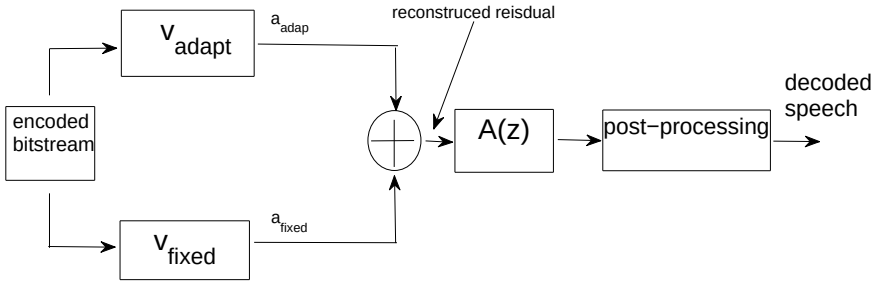

Figure 2.1: Diagram of CELP decoder.

example. We want to take advantage of this difference to detect which CELP codec has been used to process the speech signal. Thus, we need to inverse filter the decoded speech signal and extract $\mathbf{v}_{fixed}$ from the weighted sum and this requires an estimate of $\mathbf{v}_{adap}$. During encoding, $\mathbf{v}_{adap}$ is computed from the reconstructed residual of previous frames, hence it is not easy to estimate it from the decoded speech for two reasons:

1. It is difficult to accurately estimate the LPC coefficients, i.e. a_1 to a_{10}.

2. The post-processing is adaptive and we are not able to undo the process. What the post-processing is doing is dependent on the codec, but the major things are formant enhancement filtering and tilt compensation filtering. The coefficients are dependent on the speech signal so we can not perfectly invert the post-filtering.

So we chose to only use the unvoiced part of the speech signal, where the energy of $\mathbf{v}_{fixed}$ is much higher than that of $\mathbf{v}_{adap}$. A typical sentence will contain

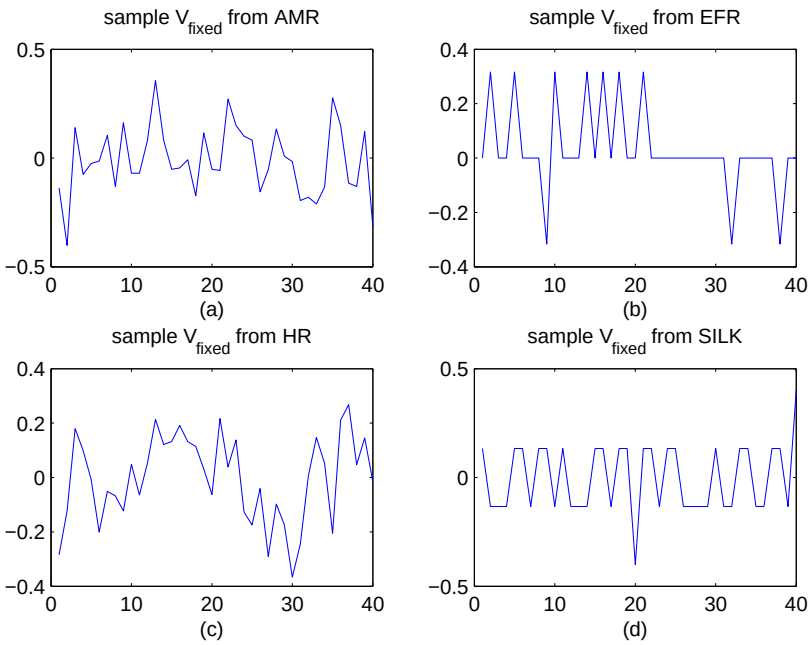

Figure 2.2: Some samples of $\mathbf{v}_{fixed}$ from different codebooks.

some unvoiced regions due to the fricatives and stop consonants.

The codebooks used for $\mathbf{v}_{fixed}$ in AMR, EFR and HR are described in [6], [7] and [8], respectively. For SILK, when operating in its low bit rate mode, $\mathbf{v}_{fixed}$ is obtained by using a pseudo-randomization algorithm algorithm whose input is a sequence of pulses similar to that shown for $\mathbf{v}_{fixed}$ in mode EFR (Fig 2.2b). As can be seen, by comparing $\mathbf{v}_{fixed}$ for SILK vs for EFR, the randomization results in a $\mathbf{v}_{fixed}$ with fewer zeros, thus making $\mathbf{v}_{fixed}$ richer. This process adds some small perturbation to the pulse sequence, and the some perturbation of constant magnitude. The sign of the perturbation is determined by the overflow-based pseudo-randomization mechanism. This process gives rise to some properties of $\mathbf{v}_{fixed}$ that we can take advantage of.

1. Due to the constant magnitude of the perturbation, the shape of $\mathbf{v}_{fixed}$ is quite restricted. The magnitude of the peaks versus the non-peaks is always 4 or 10 (see Fig 2.2d).

2. Due to the overflow based pseudo-random mechanism, some sign patterns of $\mathbf{v}_{fixed}$ appear frequently.

The improved SILK codec detection algorithm we proposed is based on these two observations.

Chapter 3

Algorithm of the Codec Detector

The codec detector can work in 4 modes: HR, EFR, AMR, and SILK. For modes HR, EFR and AMR, the framework is similar, but some part of the algorithm should be tailored to the particular codec at hand. Since mode SILK is different from the other three modes, it will be discussed in a separate subsection.

3.1 Algorithm for Mode HR, EFR and AMR

The algorithm I developed consists of the following steps.

- extract unvoiced part of speech
- linear prediction filtering of the unvoiced part of speech signal to get the residual $\mathbf{r}$
- remove $\mathbf{v}_{adap}$ from residual
- search for the best vector to fit $\mathbf{v}_{fixed}$
- get error measurement

3.1.1 Extract unvoiced part of speech

The unvoiced part of speech needs to be extracted and the same VAD algorithm is used as in the HR standard. This VAD searches for the best lag in the past

speech signal, and get the ratio between the energy in the current frame and the energy in the best-lag signal. Since it is important that we do not recognize a voiced frame as unvoiced, we use a stricter requirement on the threshold of the ratio.

3.1.2 Linear prediction filtering to get the residual

We use a 10th order linear prediction analysis, specifically the autocorrelation method, to inverse filter the unvoiced speech signal.

3.1.3 Remove $\mathbf{v}_{adap}$ from residual

In mode HR, for the unvoiced part of speech, the residual is the sum of two vectors both from $\mathbf{v}_{fixed}$. Thus, we don't have to remove $\mathbf{v}_{adap}$. In both the EFR and AMR modes, removal of $\mathbf{v}_{adap}$ is an optional step. In the unvoiced part of speech, the energy of $\mathbf{v}_{adap}$ is already very small. As such, it may not be necessary to remove this part. In fact, our experiments in the AMR mode showed that removal of $\mathbf{v}_{adap}$ degraded performance. Thus, we only performed this step in the EFR mode only.

We followed the procedure in [2] and [3] to get $\mathbf{v}_{adap}$. In the standards, the adaptive code book is the reconstructed residual of previous frames, but here we don't have these reconstructed residuals, so we use the final post-processed speech, inverse filtered with A(z), as the adaptive codebook. $\mathbf{V}_{adap}$ is obtained by searching for the best fractional lag in the adaptive code book. Even though exactly the same search algorithm and interpolation method is used as in [2] and [3], the adaptive

codebook is not very accurate. This inaccuracy doesn't affect $\mathbf{v}_{fixed}$ very much since

$$\mathbf{v}_{fixed} = \mathbf{r} - a \times \mathbf{v}_{adap} \tag{3.1}$$

in which the scaler a is often very small in the unvoiced region.

3.1.4 Search for the best vector to fit $\mathbf{v}_{fixed}$

Now that we have $\mathbf{v}_{\hat{fixed}}$, obtained by subtracting $\mathbf{v}_{adap}$ from the residual (as in EFR), or obtained without the subtraction (as in other modes), and the fixed codebook, we are ready to search for the best vector in the codebook. For every vector $\mathbf{v}$ in the codebook, we find a scaler a to minimize $\|\mathbf{v}_{\hat{fixed}} - a \times \mathbf{v}\|^2$, so we get

$$a = \mathbf{v}_{\hat{fixed}}{}^T(\mathbf{v}^T\mathbf{v})^{-1}\mathbf{v} \tag{3.2}$$

which is equivalent to find a $\mathbf{v}*$, such that,

$$\mathbf{v}* = argmin_{\mathbf{v}}(I - \mathbf{v}(\mathbf{v}^T\mathbf{v})^{-1}\mathbf{v}^T)\mathbf{v}_{\hat{fixed}} \tag{3.3}$$

3.1.5 Get error measurement

The objective function in the fitting part is normalized as our error measure.

$$err = \frac{\|\mathbf{v}_{\hat{fixed}} - \mathbf{v}*\|^2}{\|\mathbf{r}\|^2} \tag{3.4}$$

3.2 Algorithm for Mode SILK

The algorithm developed for mode SILK consists of the following steps.

- preparation of sign pattern book

- extract unvoiced part of speech
- linear prediction filtering to get the residual
- search for the most likely sign pattern
- get error measurement

3.2.1 Preparation of sign pattern codebook

As mentioned in Chap. 2, the sign pattern of $\mathbf{v}_{fixed}$ is very limited in SILK and we want to use this sparsity to detect if SILK was used. The first step in this process is to build a representative and efficient sign pattern codebook. To do an exhaustive search over all possible sign patterns is impractical and we can reduce the search space by answering the following two questions.

1. Do we need the length of sign pattern to be the frame length?
2. What are the most frequent sign patterns? Every pulse sequence p may have a different $\mathbf{v}_{fixed}$ sign pattern, and the number of pulses in one frame is not fixed, even in the low bitrate mode of SILK. Thus, when we construct the sign pattern book, enumerating all possible sign patterns can be inefficient.

To help us answer these 2 questions, let's define a binary pattern BP, for 10 consecutive samples from $\mathbf{v}_{fixed}$,

$$BP = \sum_{p=1}^{10} 2^{p-1} s(p) \tag{3.5}$$

$$s(p) = \begin{cases} 1 & : \quad if\ pth\ sample\ is\ positive \\ 0 & : \quad if\ pth\ sample\ is\ negative \end{cases} \tag{3.6}$$

BP is just a way to describe a sign pattern using a number. Figure 3.1 shows the histogram of BP for a SILK compressed speech sentence. We can see several peaks in the histogram of the first 10 samples of each frame. There are fewer and smaller peaks with the second and third histograms. Finally, the fourth histogram is almost flat. As it goes along the sequence, the sign pattern becomes more and more random. In our sign pattern codebook, the first 30 samples of a frame are included. Thus our first question is answered.

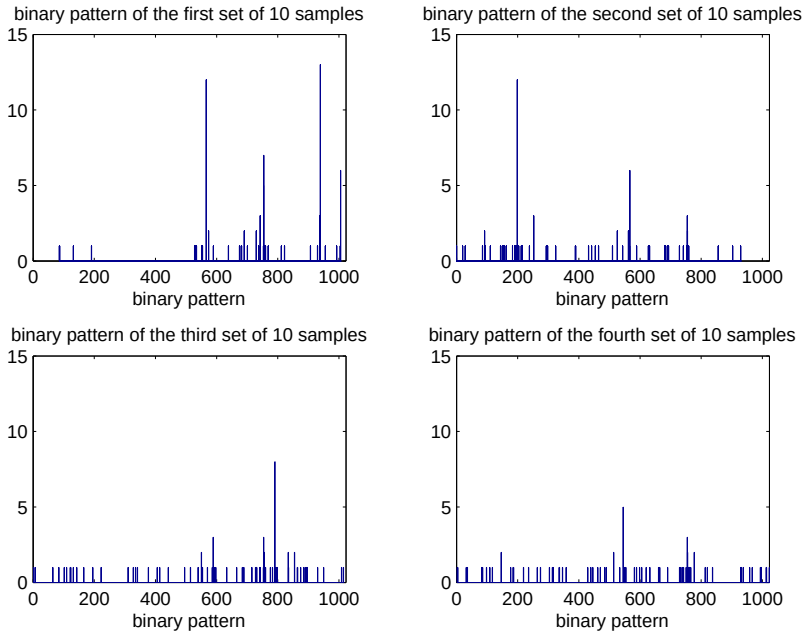

Figure 3.1: Histogram of binary pattern of a SILK sentence.

Another observation is, for every 10 positions in a frame, the number of nonzero pulses is less than 3 most of the time. So the sign pattern book is designed to include

just the sign patterns generated by these pulses. This is the answer to our second question.

3.2.2 Search for the most likely sign pattern

Denote $\mathbf{v}_{short}$ as a vector containing the first 30 samples of $\mathbf{v}_{fixed}$. For every sign pattern $\mathbf{s}$ in the codebook, we search for the $\hat{\mathbf{s}}$ which maximizes the correlation,

$$corr = \mathbf{s}^T \mathbf{v}_{short} / (\|\mathbf{s}\| \|\mathbf{v}_{short}\|) \tag{3.7}$$

3.2.3 Get error measurement

The error is defined as the number of inconsistent signs between $\mathbf{v}_{short}$ and $\hat{\mathbf{s}}$.

$$err = \sum_{i=1}^{30} sign(i) \tag{3.8}$$

$$sign(i) = \begin{cases} 1 & : \quad if \;\; \mathbf{v}_{short}(i) \times \hat{\mathbf{s}}(i) < 0 \\ 0 & : \quad if \;\; \mathbf{v}_{short}(i) \times \hat{\mathbf{s}}(i) \geq 0 \end{cases} \tag{3.9}$$

3.3 Experiments of Codec Detector

We took 100 sentences from the TIMIT database that were recorded using 10 different microphones. The sentences contain speech from both male and female speakers. For every sentence, we encoded and decoded using GSM-HR, GSM-EFR, GSM-AMR(5.9kb mode) and SILK. This process resulted in 5 datasets, $data_{origin}$, $data_{AMR}$, $data_{EFR}$, $data_{HR}$, and $data_{SILK}$. A total of 81480 frames (16296 frames in 100 sentences $\times$ (4 modes + original)) are used, about 27 minutes, of which approximately one third are unvoiced. We ran the detector in its 4 modes on every

dataset. Thus each time the detector is asked if the frame is previously compressed by, for example, HR, and it tells us the result for all the sentences in the 5 datasets.

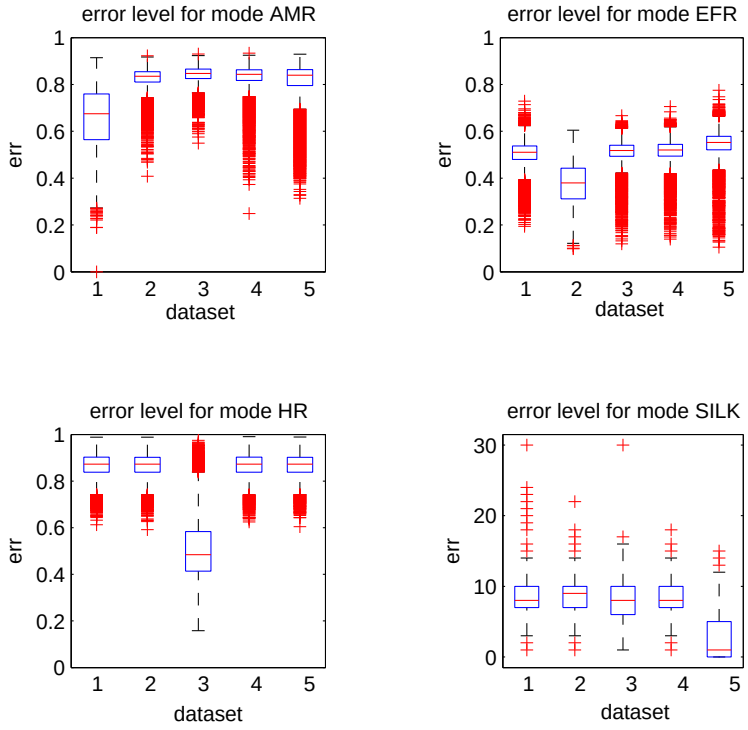

Figure 3.2: dataset 1 is for AMR coded speech, dataset 2 is EFR coded, dataset 3 is HR coded, dataset 4 is for the original speech, and dataset 5 is SILK encoded.

Figure 3.2 shows the error distribution for the detectors. Every plot corresponds to one of the 4 modes, and in each plot, every box represents one of the 5 datasets. The lines at the center of the box are the mean normalized error, the edges of the box are the 25th and 75th percentiles, and the red crosses are outliers. As we can see from Figure 3.2, for a specific mode, the error is the lowest for the data using that codec. Now we are ready to answer the question that, given a speech file,

which codec is used to generate it.

We model every box (one dataset for each mode) in Figure 3.2 as a Gaussian, and get the likelihood ratio for every mode. For example, in mode AMR, we model err_{AMR} as AMR Gaussian and err_{EFR}, err_{HR}, err_{SILK} and err_{origin} together as non-AMR Gaussian. For an unknown speech sentence s, we run the algorithm in mode AMR, get the log likelihood ratio using AMR Gaussian and non-AMR Gaussian. We have 4 modes so we'll have 4 likelihood ratios, llr_{AMR}, llr_{EFR}, llr_{HR}, llr_{SILK}. Using these 4 as the input for a logistic regression classifier, we get our final classification result.

We used 80% sentences in the dataset to get the Gaussian parameters and we used the rest to do the test.

Table 3.1 confusion matrix for codec detector on TIMIT dataset

classified as	AMR	EFR	HR	SILK	origin
AMR	**89.89**%	0.48%	0.06%	5.00%	4.58%
EFR	0.18%	**91.27**%	0.60%	0.00%	7.95%
HR	0.06%	1.81%	**96.51**%	0.06%	1.57%
SILK	7.89%	0.60%	0.12%	**73.99**%	17.40%
ORIGIN	1.81%	3.55%	0.42%	6.44%	**87.78**%

Chapter 4

An Improved Algorithm for SILK

As shown in the previous chapter, the frame level detection rate for SILK is around 70% while the others are around 90%. There are two shortcomings of the previous algorithm. First, it doesn't take into account the magnitude pattern of $\mathbf{v}_{fixed}$ from SILK, as mentioned in Chap. 2 (page 5). Second, the sign pattern of the residual $\mathbf{r}$ may not behave the same as the standard describes, due to

1. The post-process in the SILK decoder.
2. The voiced/unvoiced decision may not be the same as in the first encoding process.
3. The linear predictive coefficients are not exactly the same as in the first encoding process, even though that the same LP analysis is used.

To overcome these shortcomings, we built a new codebook that accounts for these artifacts. Since we already have the SILK codec, we encode and decode a training $dataset_{origin}$ to get $dataset_{SILK}$, and perform our voiced/unvoiced decision and linear prediction analysis on $dataset_{SILK}$. Then we use all of the residuals to build the new codebook. Thus, our new algorithm is partly knowledge-based and partly data-driven as will be described in detail in the next section.

4.1 Training

Figure 4.1 shows a diagram for the improved algorithm. In the training step, we take a large amount of speech sentences, encode and decode them using SILK, do the voiced/unvoiced decision and LP analysis to get all the normalized residuals $\mathbf{r}$. We keep the first 30 samples of all the residuals $\mathbf{r}$ to construct a codebook, cb_{short}, plus 3 distributions based on the 2 observations mentioned in Chap. 2 (page 5).

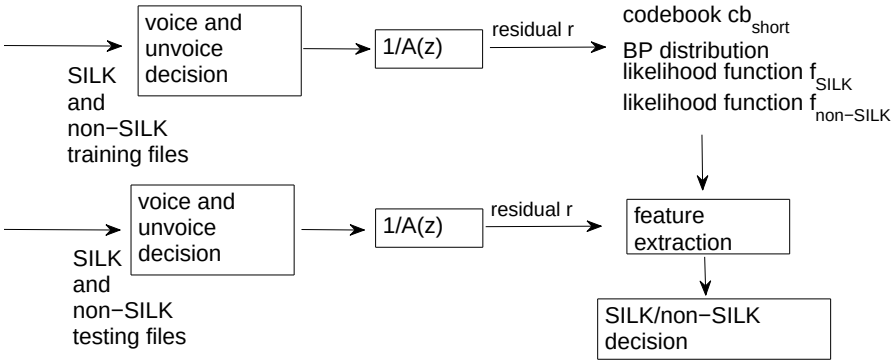

Figure 4.1: diagram of the improved algorithm.

The first distribution for BP, which describes the sign patterns and we model it as multinomially distributed.

The second and the third distributions are based on the observation 1 on page 5. The magnitude of the peaks versus the non-peaks is always 4 or 10 (see Fig 2.2d). So if the peaks are removed, we expect the variance of the rest signal to be very small. For a given normalized residual $\mathbf{r}$, we remove all the peaks to get $\mathbf{r}'$, so the length of $\mathbf{r}'$ is less than $\mathbf{r}$. We define the non-peak variance, $V_{non-peak}(\mathbf{r})$, as the variance of $\mathbf{r}'$. We have two training datasets, training $dataset_{SILK}$ and

training $dataset_{non-SILK}$. We partition $V_{non-peak}$ into 460 bins and model $V_{non-peak}$ for each dataset as multinomially distributed. We didn't model it as normally distributed since we observed that the $V_{non-peak}$ for training $dataset_{non-SILK}$ is highly unbalanced, or heavy tailed, as shown in Figure 4.2.

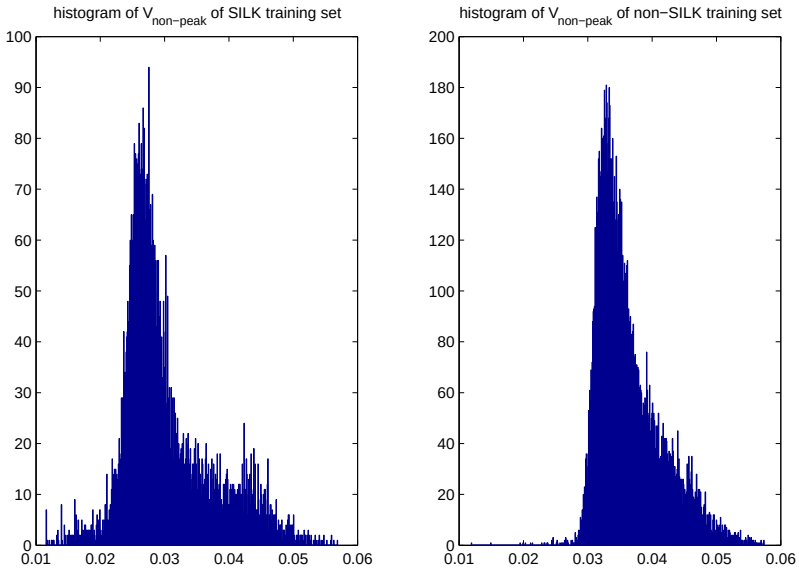

Figure 4.2: histogram of $V_{non-peak}$ for both SILK and non-SILK files.

4.2 Testing

For a given normalized residual $\mathbf{r}$ from the test dataset, we extract three features $\mathbf{feature}(\mathbf{r})$.

- $feature_1(\mathbf{r}) = Pr(BP(\mathbf{r}))$.
- $feature_2(\mathbf{r})$ = the number of inconsistent signs between r(1:30) and r', where r' is the value in cb_{short} that has the highest correlation with r(1:30).

- $feature_3(\mathbf{r}) = \log \frac{f_{SILK}(V_{non-peak}(\mathbf{r}))}{f_{non-SILK}(V_{non-peak}(\mathbf{r}))}$ is the log likelihood ratio.

We classify the frame as non-SILK only when $feature_1(\mathbf{r}) = 0$ and $feature_2(\mathbf{r}) < threshold_1$ and $feature_3(\mathbf{r}) < threshold_2$, where $threshold_1$ and $threshold_2$ are optimized using the training set.

4.3 Experiments of SILK codec detector

We took 100 sentences from TIMIT, with gender balanced, and one third of the frames classified as unvoiced. The sampling frequency is 16kHz. We encoded and decoded these 100 sentences using SILK, and the target bitrate was 10kbps. This is our training $dataset_{SILK}$. For training $dataset_{non-SILK}$, we encode these same 100 sentences using AMR-WB[11], and the target bitrate is 12.65kbps, together with the original timit files. So training $dataset_{non-SILK}$ is twice as large as training $dataset_{SILK}$. The test set contains another 100 sentences from timit database, processed in the same way.

We also did a MFCC-GMM, fully data-driven baseline for comparison. To obtain the MFCCs, the window size was 20ms, and we added both delta and delta-delta coefficients, so a total of 39 cepstral coefficients for each 20 ms, with no overlap. We used the EM algorithm to train two GMM, each with 256 Gaussians, one for SILK and the other for non-SILK. We used the log-likelihood ratio to make the decision. Table 4.1 shows the frame-level result.

Table 4.1 results for proposed algorithm and MFCC-GMM baseline

	detection rate	false alarm
proposed algorithm	81.50%	18.50%
MFCC-GMM baseline	69.98%	36.64%

In Table 4.2, we give results that show the robustness to high-pass and low-passing filtering of the proposed algorithm.

Table 4.2 experiments of robustness under various filtering

	HP $F_{stop} = 100Hz$	HP $F_{stop} = 200Hz$
detection rate	63.09%	59.40%
	LP $F_{stop} = 7900Hz$	LP $F_{stop} = 7800Hz$
detection rate	60.21%	58.30%

Chapter 5

Application to Tampering Detection

5.1 Tampering Detection Algorithm

Here we propose a tampering detection algorithm based on the codec detector. For a speech file that has already been encoded and decoded by a specific codec, our goal is to tell if someone else has deleted or inserted anything after its generation. We observed during our experiment that the proposed algorithm is very sensitive to shifting. The basic idea is to shift the frame by throwing away samples in the beginning. In the case where we throw away correct number of samples, the frame grid will be the same as in the encoding process, and as a result, we should see certain property as will be described later.

For a speech sentence we extract all the unvoiced parts. Say we have a total of N unvoiced frames, then we divide them into M segments, so every speech segment has $k = \frac{N}{M}$ unvoiced frames, and k is defined as the unit size. For every segment, we detect the subframe offset, so if two consecutive segments have inconsistent offsets, we claim that the later segment has been tampered. We want the unit length to be as small as possible since we can locate the tampering more accurately. Next we introduce how to detect the subframe offset for a segment.

for i = 1 : framelength **do**

throw away first $i - 1$ samples and run the algorithm in mode m for k frames,

so we have k errors, $err_{1:k}$.

$m(i) = mean(err(1:k))$

end for

offset for this segment is the index of the first dip in m.

The tampering detection algorithm here is only for the codec detector in Chap. 3 for clarity. For the improved SILK detector, we need to get $feature_1(1:k)$ instead of $err_{1:k}$, and to claim that the offset is the index of the first peak in m.

5.2 Experiments on Real Cellphone Recordings

We tested the algorithm on a cellular dataset. This dataset consists of recordings directly from cellphone conversations, where they may undergo channel distortion and a cellphone enhancement process. We know nothing about what coding standard was used, except that it's within the GSM family. The beginning parts of the speech signals were chopped.

Figure 5.1 shows the mean curve for a segment from cellular and as a reference, a segment from a microphone recording that did not undergone any speech coding process. We can observe in the line marked with circles a dip every 40 samples, which is the subframe length. The subframe offset is 20, since the first dip appears at offset 20.

Then we did some experiments on 50 sentences from the cellular database, each of them was about 5 minutes long, yielding a total of 250 minutes. We ran the tampering detection algorithm to get the mean curves for all the unvoiced subframes.

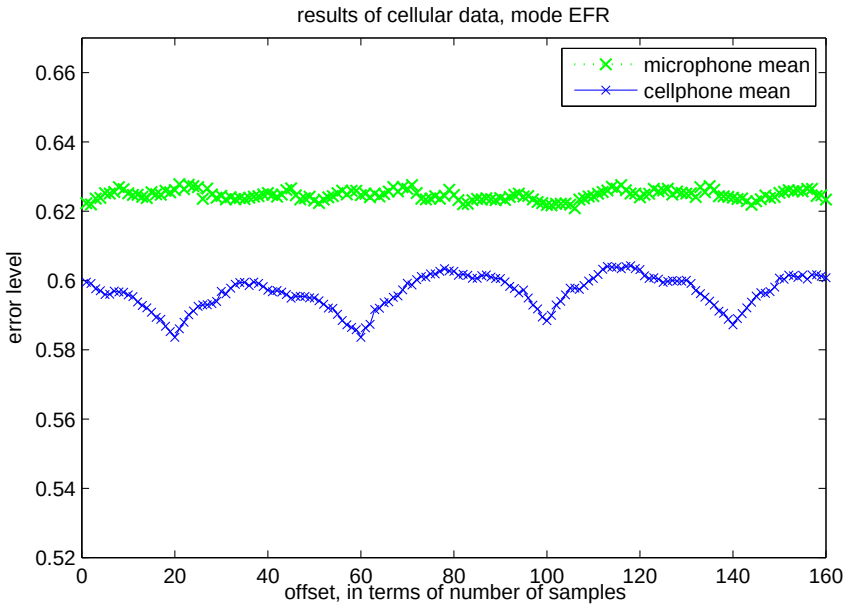

Figure 5.1: Mean curve for one segment from cellular database.

For each of them, we used a convex hull to locate all the dips and if the distance between all the dips are within some tolerance range, we claim the first dip location as the offset for the segment, otherwise we claim no offset can be detected. We detected 5813 offsets for 11797 unvoiced segments. Note here the cellphones may not operate in mode EFR all the time and we don't know the ground truth.

5.3 Tampering Experiments

We also did an experiment on tampering detection. We took a timit sentence $\mathbf{t}_1$ "They had come not to admire but to observe", encoded and decoded using SILK to get $\mathbf{t}_2$, then we did tampering on $\mathbf{t}_2$, deleting the word "not", to get sentence $\mathbf{t}_3$. We ran the proposed tampering detection algorithm on $\mathbf{t}_3$ to get the offset track, as

shown in Figure 5.2.

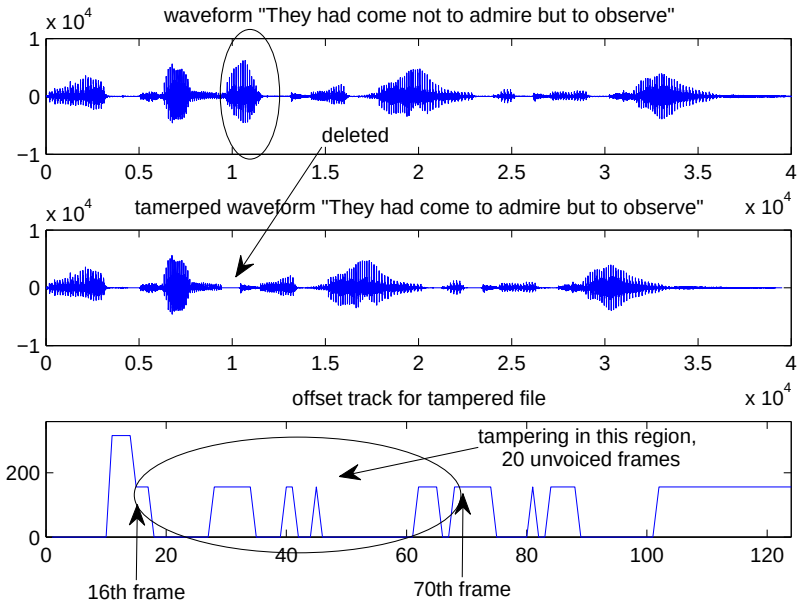

Figure 5.2: top: waveform of a SILK encoded sentence. middle: tampered waveform from the top. bottom: offset track for the tampered waveform.

The offset is zero for voiced frames and the maximum value in this case is 320 since the length of a 20ms frame in 16kHz is 320, different from the length in the cellular database. We can see there is an inconsistency in the offset track, i.e., drops from 316 to 156 at the 16th frame. around the tampering spot. This offset corresponds to the tampering spot since the offset is assigned for each segment and each segment is 20 unvoiced frames. Taking these facts into considertion, we claim the tampering occurs between 16th frame and 70th frame, which is between 5121th sample and 22400th sample.

Bibliography

[1] D. Garcia-Romero and C. Y. Espy-Wilson, "Automatic Acquisition Device Identification from Speech Recordings", in International Conference on Acoustics, Speech and Signal Processing (ICASSP), Dallas, TX, USA, 2010.

[2] Scholz,K.,Leutelt,L.,Heute,U., "Speech-Codec Detection by Spectral Harmonic-Plus-Noise Decomposition", Systems and Computers, 2295 - 2299 Vol.2 2004.

[3] Yang,R.,Qu,Z.,Huang,J.,"Detecting Digital Audio Forgeries by Checking Frame Offsets", MM and Sec08,September 22C23, 2008, Oxford, United Kingdom.

[4] H. Farid,"Detecting Digital Forgeries Using Bispectral Analysis",MIT AI Memo AIM-1657, MIT,1999.

[5] C. Grigoras,"Digital Audio Recording Analysis: The Electric Network Frequency (ENF) Criterion.", The International Journal of Speech Language and the Law, vol.12, no.1, pp.63-76, 2005.

[6] "ETSI GSM 06.90 Digital cellular telecommunications system (Phase 2+); Adaptive Multi-Rate (AMR) speech transcoding", 2000.

[7] "ETSI GSM 06.60 Digital cellular telecommunications system (Phase 2+); Enhanced Full Rate (EFR) speech transcoding", 1999.

[8] "ETSI GSM 06.20 Digital cellular telecommunications system (Phase 2+); Half Rate (HR) speech; Part 2: Half rate speech transcoding", 1998.

[9] SILK Speech Codec draft-vos-silk-02, www.ietf.org/id/draft-vos-silk-02.txt

[10] Spanias,A., "Speech coding: a tutorial review", Proceedings of the IEEE, volume 82 issue 10 pp 1541 - 1582, 1994.

[11] "Wideband coding of speech at around 16 kbit/s using Adaptive Multi-Rate Wideband (AMR-WB)", 2008