Automatic Speech Codec Identification with Applications to Tampering Detection of Speech Recordings

Jingting Zhou, Daniel Garcia-Romero, Carol Espy-Wilson

Department of Electrical and Computer Engineering, University of Maryland, College Park, MD

jingtingzhou@hotmail.com, dgromero@umd.edu, espy@umd.edu

10.21437/Interspeech.2011-648

Abstract

In this paper we explored many versions of CELP codecs and studied different codebooks they use to encode noisy part of residual. Taking advantage of noise patterns they generated, an algorithm was proposed to detect GSM-AMR,EFR,HR and SILK codecs. Then it's extended to identify subframe offset to do tampering detection of cellphone speech recordings.
Index Terms: forensic,compression,CELP

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.

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 [2,3,4,5].

Scholz [1] 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 SVM, support vector machine, classifier to determine which of five different codecs was used. Yang [7] examined the compression mechanism of mp3 files, when an mp3 file is encoded, the audio samples are divided into frames, each frame has its own frame offset after encoding. By examining the trace left by quantization process, the frame offset can be detected with high accuracy. Forgeries will break the original frame grids, thus leave us evidence.

In this paper, we studied various speech codecs in today's digital communication system. By examining different noise patterns generated in each codec, we proposed a framework to detect 4 different codecs, GSM-Adaptive Multi-Rate(AMR) [2] in 5.9kbps, GSM-Enhanced Full Rate(EFR) [3] in 12.2kbps, GSM-Half Rate(HR)[4] in 5.6kbps, and SILK [5](speech codec used in Skype) in 5kbps.

2. CELP Family of Speech Codecs

Spanias [6] 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.

Fig.1 shows a block diagram of the decoding process of a CELP codec. 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. V_{adap} is a vector copied from an adaptive codebook and it contains previous reconstructed residuals, and it captures periodic portions so the energy is high in voiced regions. The weighted sum of these two vectors, reconstructed residual r, is fed into the inverse LPC filter. The corresponding weights for V_{adap} and 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 V_{fixed}. We want to take advantage of this difference to detect which CELP codec has been used on the speech signal. Thus, we need to extract V_{fixed} from the weighted sum and this requires an estimate of V_{adap}.During encoding, V_{adap} is computed from the reconstructed residual of previous frames, hence it is not easy to estimate from the decoded speech. Mostly because:

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 V_{fixed} is much higher than that of V_{adap}. A typical sentence will contain some unvoiced regions due to the fricatives and stop consonants.

The codebooks used for V_{fixed} in AMR, EFR and HR are described in [2], [3] and [4]. For SILK, in its low-bitrate modes, normally two to three pulses are encoded per subframe, each having an amplitude of +1 or -1. Since the pulse sequence is

Copyright © 2011 ISCA

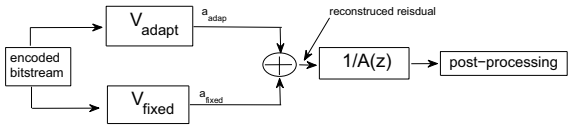

Figure 1: *diagram of CELP decoder.*

too sparse to be used as V_{fixed}, an interesting mechanism was introduced in [5] to convert a pulse sequence into V_{fixed}. After adding a small amount of offset to the pulse sequence, an overflow-based pseudo random sequence is generated according to sequence, and the sign of V_{fixed} may be flipped based on the pseudo random sequence.

Since V_{fixed} is generated in this particular way, we can show that although there might be many possibilities of the sign pattern of the pulse sequence, the sign pattern of V_{fixed} is much more limited. This limitation can be used to identify if a frame has been compressed by SILK, as will be explained in the following section.

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. Mode SILK is different from the other 3 modes, so it will be discussed in a separate subsection.

3.1. Algorithm for Mode HR, EFR and AMR

3.1.1. Algorithm outline

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

3.1.2. Extract unvoiced part of speech

The unvoiced part of speech needs to be extracted since our goal is to see how closely V_{fixed} can be represented by a particular codebook. We want the voiced/unvoiced partition as close to the original partition in the compression process as possible. Consequently, the same voiced/unvoiced decision algorithm is used as in the HR standard. Since it is important that we do not recognize a voiced frame as unvoiced, we use a stricter requirement than the standard.

3.1.3. Linear prediction filtering to get the residual

Here a 10 order linear prediction analysis is performed, and the autocorrelation method is used.

3.1.4. Remove V_{adap} from residual

In mode HR, for the unvoiced part of speech, the residual is the sum of two vectors both from V_{fixed}. Thus, we don't have to remove V_{adap}. In both the EFR and AMR modes, removal of V_{adap} is an optional step. In the unvoiced part of speech, the energy of 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 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 the V_{adap}, except that final post-processed speech is filtered with A(z) as the adaptive codebook for the next frame. Even though exactly the same search algorithm and interpolation method is used as in [2] and [3], the adaptive codebook is not very accurate. Subtracting V_{adap} from the residual should give us $\hat{V_{fixed}}$.

3.1.5. Search for the best vector to fit V_{fixed}

Now that we know $\hat{V_{fixed}}$ and the fixed codebook, we are ready to search for the best fitted vector in the codebook. For every vector $\mathbf{v}$ in the codebook, we find a gain $\mathbf{v}*$ that minimizes

$$\mathbf{v}* = argmin_{\mathbf{v}}(I - \mathbf{v}(\mathbf{v}^T\mathbf{v})^{-1}\mathbf{v}^T)V_{fixed} \quad (1)$$

3.1.6. Get error measurement

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

$$err = \frac{\|V_{fixed} - \mathbf{v}\|^2}{\|\mathbf{r}\|^2} \quad (2)$$

3.2. Algorithm for Mode SILK

3.2.1. Algorithm outline

- 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

3.2.2. Preparation of sign pattern codebook

As mentioned in section II, the sign pattern of V_{fixed} is very limited in SILK and we want to use this sparsity to detect if the 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 sign pattern to be of length 160? The pseudo random sequence is generated every frame, and V_{fixed} is of length 160.
2. What are the most frequent sign patterns? Every pulse sequence p may have a different 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 V_{fixed},

$$BP = \sum_{p=1}^{10} 2^{p-1}s(p) \quad (3)$$

$$s(p) = \begin{cases} 1 & : \ if\ pth\ sample\ is\ positive \\ 0 & : \ if\ pth\ sample\ is\ negative \end{cases} \quad (4)$$

BP is just a way to describe a sign pattern using a number. Fig. 2 shows the histogram of BP for a SILK compressed

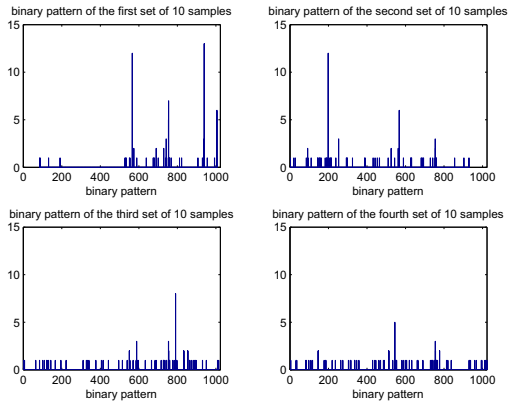

Figure 2: *histogram of binary pattern of a SILK sentence.*

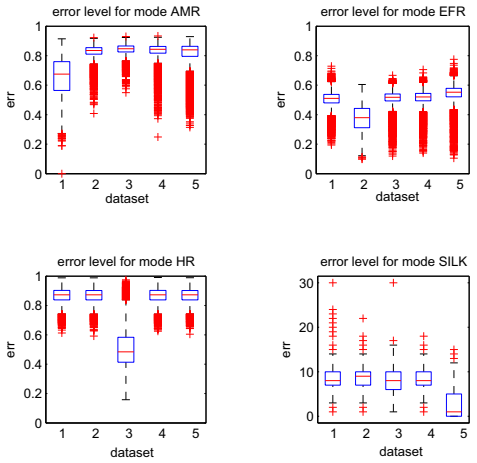

Figure 3: *Error Level for the whole 5 datasets: dataset 1 is for AMR coded speech, dataset 2 is EFR coded, dataset 3 is HR coded, dataset 4 is for the original speech, and dataset 5 is SILK encoded.*

speech sentence. We can see several peaks in the histogram of the first 10 samples. 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.

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.

3.2.3. Search for the most likely sign pattern

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

$$corr = \mathbf{s}^T V_{short} / (\|\mathbf{s}\| \|V_{short}\|) \tag{5}$$

3.2.4. Get error measurement

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

$$err = \sum_{i=1}^{30} sign(i) \tag{6}$$

$$sign(i) = \begin{cases} 1 & : \quad if \quad V_{short}(i) \times \hat{\mathbf{s}}(i) < 0 \\ 0 & : \quad if \quad V_{short}(i) \times \hat{\mathbf{s}}(i) \geq 0 \end{cases} \tag{7}$$

3.3. Experiments of Codec Detector

We took speech sentences from the TIMIT database, 100 sentences, from 10 different microphones, and it includes both male and female speakers. For every sentence, we encoded and decoded using GSM-HR, GSM-EFR, GSM-AMR(5.9kb mode) and SILK. So we now have 5 kinds of dataset, $data_{origin}$, $data_{AMR}$, $data_{EFR}$, $data_{HR}$, and $data_{SILK}$. A total of 16296×5 frames are used, about 27 minutes, of which approximately one third are unvoiced. We ran the detector in its 4 modes on every dataset, which means 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.

Fig.3 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 Fig.3, for a specific mode, error is the lowest for the data using that codec. A threshold is set for every mode to tell if a specific codec is used, i.e, a threshold for each plot in Fig.3. HR has the best performance with a detection rate around 93% and a false alarm rate around 1%. EFR and AMR have detection rates around 80% and a false alarm rate around 5% and 10%, respectively. Keep in mind this is frame by frame performance and for a speech sentence, we can always combine all the unvoiced frames and take the majority votes as a final result. During the experiment, the detection rate is 100% at the sentence level, when we use majority vote.

For mode SILK, the performance is not as good as the other three, with detection rate around 80% and false alarm around 20%. But during the experiment, we found that for $data_{origin}$, $data_{AMR}$, $data_{EFR}$, $data_{HR}$, none of the frames have zero error (error in SILK is defined in section 3.2.5). In other words, there is no perfectly consistent sign pattern in those datasets. For the 100 SILK compressed speech sentences, there are always some portion of the unvoiced frames that have a perfectly matching sign pattern. So taking advantage of this observation, we can also achieve a 100% detection rate in SILK at the sentence level.

4. Applications to Tampering Detection

In this section we introduce a tampering detection algorithm. For real world scenarios, we often encounter files whose beginning parts have been chopped so we don't know the original frame grid. So for them, we had to not only detect the coder, but also the original frame grid. We first introduce the tampering detection algorithm and then show the results in a subsection.

4.1. Tampering Detection Algorithm

For a short speech segment, we define the original index in the original subframe grid of the first sample as the subframe offset. If nothing is done, the subframe offset is 1. If the speech is chopped in a way that first n samples are thrown away, the subframe offset for this segment is n+1, since the first sample in the chopped file is the $(n+1)^{th}$ originally.

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 segements, so every speech segment has $k = \frac{N}{M}$ unvoiced frames, and k is defined as unit size. For every segment, we detect the subframe offset, so if two consecutive segments have inconsistent offsets, we claim there's tampering in the later segment. 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 subframe offset for a segment.

4.1.1. Subframe offset detection algorithm

The basic idea is that if, after throwing away the first n samples, the resulting subframe grid is the same as the original subframe grid, then the error will be very low. Here's the way to detect subframe offset, the unit size is k.
for offset = 1 : 160

- throw away first $(offset-1)$ samples and run the algorithm in mode m for k frames, so we have k errors, $err_{1:k}$.
- $m_{err}(offset) = mean(err)$.

end
If 3 or 4 dips can be observed in the m_{err} curve, and the index difference between two dips is 40, i.e. one dip is at offset i and the next dip is at offset $i+40$, we claim the index of the first dip as the subframe offset.

During the experiment, we found that for the k errors, the lowest extreme data point that were not considered outliers (denote it as ENO for short), is more sensitive to offset than mean, so it's also included. Of course, the offset detection algorithm will work only if the codec detector is working in the correct mode, otherwise we can not find any dip.

4.2. Experiment on Real Cellphone Data and Detection of Subframe Offset

We tested the algorithm on a cellular dataset. These are 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.

Fig.4 shows the mean curve and ENO curve for a segment from cellular and a segment from a microphone recording. The microphone recording is for comparison, which has not undergone any speech coding process. We can observe in the solid line a dip every 40 samples, i.e. the subframe length. The subframe offset is 20, since the first dip appears at offset 20. We've run 5 files from the cellular database and we observed a similar dip pattern. The codec detector was working in mode EFR here.

We found a unit size of 200 is enough and the algorithm is robust to filtering, but not to additive noise (both filtering and additive noise applied to the decoded signal). It is not surprising since what we are taking advantage of is the codec noise pattern. Once the shape of the noise is destroyed by additive noise, the algorithm stops working. Accuracy and robustness is often a trade off in a forensic task, so in future work, long term dependence and properties of the voiced part the speech signal should be explored to see if it can improve robustness.

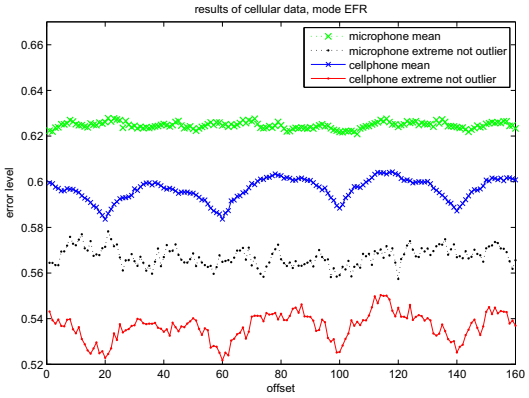

Figure 4: *mean and ENO error for every offset,both for cellular file and microphone file.*

5. Conclusion

In this paper we proposed an algorithm to do media authentication in two steps. 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. We focused on codecs in CELP family and we just use the unvoiced part of speech. We are planning to extend to voiced part and outside CELP family in future work.

6. References

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

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

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

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

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

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

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