Multimedia Systems assignment
GSM 06.10 vocodec

Thodoris Tyrovouzis 9369

March 7, 2022

1 Introduction

This document provides some general guidance for the codec's source code. My goal in this assignment was to follow the specification up to **Παραδοτέο Επιπέδου 3** as closely as possible, using simple and readable code. Towards the end of the report, I also discuss alternate implementations to potentially improve the audio quality. The source code is hosted on GitHub at `https://github.com/Thodoris1999/GSMVocodec`.

2 Source code description

The source code contains the following utility functions.

packFrmBitStrm, unpackFrmBitStrm

Pack and unpack coded frame packets to and from the frame parameters.

lar, lar_inv

Transform LPC reflection coefficients to and from log area ratios

quantize_lars, dequantize_lars

Quantize/Dequantize log-area ratios.

LTP_gain_code, LTP_gain_decode

Code and decode long term prediction gains b_j.

preproc, postproc

Preprocess and postprocess procedures (paragraphs 3.1.1, 3.1.2, 3.2.4 of the standard).

acf

Estimates the autocorrelation function from the samples.

block_weight_filter

Applies the block weighting filter specified in the RPE encoding section.

RPE_decimation_selection

Downsample LTP residuals with grid selection

quantize_xprime, dequantize_xprime

Quantize/Dequantize LTP residuals.

ST_filtering

Short term analysis filtering (with interpolated log-area ratios)

ST_synthesis

Short term synthesis filtering (with interpolated log-area ratios)

The rest of the functions, **RPE_frame_coder**, **RPE_frame_decoder**, **RPE_frame_ST_coder**, **RPE_frame_ST_decoder**, **RPE_frame_SLT_coder**, **RPE_frame_SLT_decoder** are implemented as described in the assignment description.

The main function to test the codec is the **encode_audio(file)** function. It takes an audio file as an argument and plays its audio after it has been coded and decoded. In the project folder, there is already a speech sample, so the project can be tested by running.

```
encode_wav('OSR_us_000_0010_8k.wav')
```

or just **demo** on the MATLAB interpreter.

3 Implementation notes

The project repository has 3 branches. The main branch contains the 3rd level assignment which follows the implementation specified in the standard as closely as possible, including the interpolation of log-area ratios. The level3_bwe branch targets the 3rd level as well, but uses bandwidth expansion. The stage2 branch can be used to verify that the first two levels are working correctly.

Initially, I attempted to follow the specification on the assignment description as closely as possible. However, without any modifications, occasional high frequency crackling noise could be heard in any speech sample I tried. After analyzing, I found out that this happened due to sudden spikes towards the end of frames in which the transfer function of the short term synthesis filter was unstable (or had poles outside the unit circle). Theoretically, this is still not an issue when applying the synthesis filter on the original residuals, but small errors due to floating point operation precision and quantization can produce divergent behavior[1].

It turned out that it was important for that matter to interpolate the log-area ratios. After doing this, the entire frame is not fed to the same filter and therefore in practice never escalated to divergence, fixing the crackling issue completely. To implement this however, I needed to slightly change the function signatures of the spec in the assignment description to include the previous log-area ratios.

Another commonly used[2] way to fix the synthesis instability, was to introduce Bandwidth expansion[1] on the decoder's synthesis filter. Of course, this distorts the decoded signal slightly, as the synthesis filter is not exactly inverse of the analysis filter, but it is not that noticeable. It also does not make the codec depend on the previous log-area ratios, allowing me to follow exactly the function spec of the assignment description. In my opinion, this also sounds a bit more clear than the LAR interpolation approach.

The implementation in the stage2 branch contains only the first two levels, so no coding/quantization of the long term analysis residuals is applied. This produces results almost identical to the original audibly, which makes sense since only the LARs and the LTP gain are compressed while the residuals remain equal to the original and the majority of the compression occurs on the residuals (instead of 160 floats we only use 188 bits in each frame for the RPE parameters!). Bandwidth expansion can be optionally enabled by uncommenting it from `RPE_frame_ST_decoder.m`, but the high frequency spikes are not as prevalent since we are using the original LTP residuals. Since both the first and secod level assignments need to be working correctly, this branch can be used to test them.

References

[1] Dominik Roblek et al. "Real-time Speech Frequency Bandwidth Extension". In: *2021 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2021. URL: `https://arxiv.org/abs/2010.10677`.

[1]`https://www.vocal.com/speech-coders/stability-of-lpc-filters/`
[2]`https://patents.google.com/patent/WO2001003124A1/ru`