Introducing New Features in (Lossless) Audio Compression

Maribor, 14. 1. 2025

David Podgorelec

University of Maribor
Faculty of Electrical Engineering and Computer Science

Institute of Computer Science
Laboratory for Geospatial Modelling, Multimedia and Artificial Intelligence

Prediction-based lossless compression

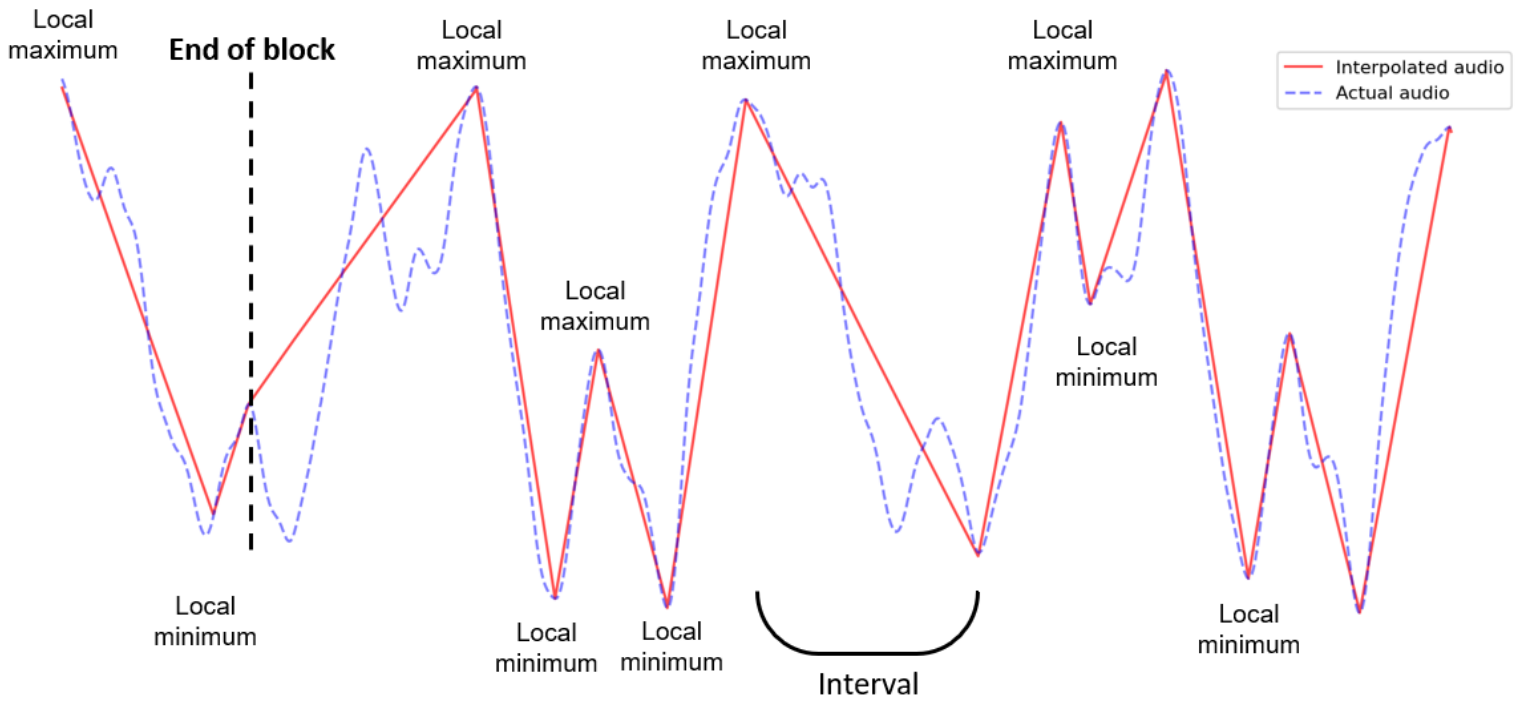
- Sample-based predictions
  - Each sample predicted from its recent history.
    a) Single prediction type → encode only residuals (and a few starting samples).
    b) Different prediction types → encode types (with short codes) and residuals.
- Feature-based predictions
  - The same prediction type for the whole segment.
  - Prediction (feature) interpolates (approximates) the segment.
  - Each segment coded by:
    - Feature type (except there is a single type)
    - Feature parameters (segment borders + interpolation function)
    - Residuals.

Feature-based predictions

- Feature parameters coding = lossy compression of a segment
- All features → lossy compression of the whole input stream
- Is this good (useful)? Yes, if:
  - Compact representation of (all) features.
  - Small residuals mostly (low entropy, good „compressibility“).
  - Note that features and residuals are losslessly compressed afterwards.

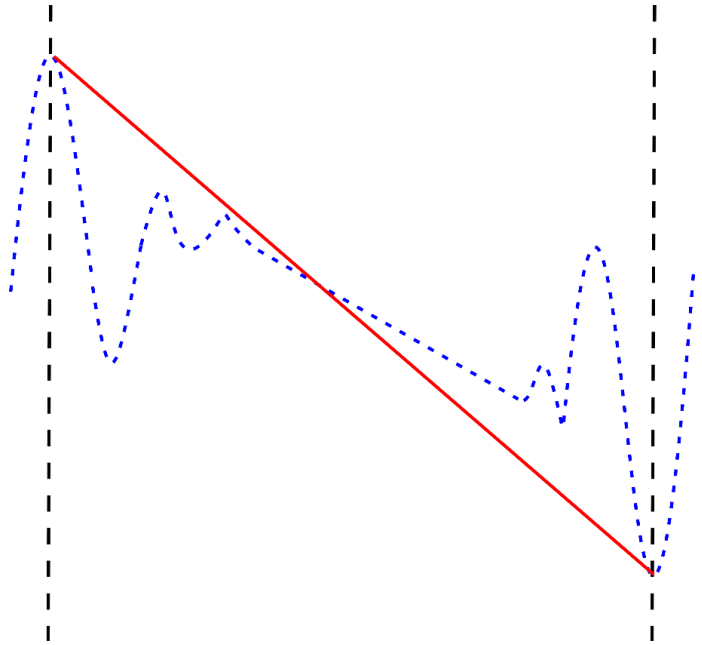
Feature-based predictions (our experience)

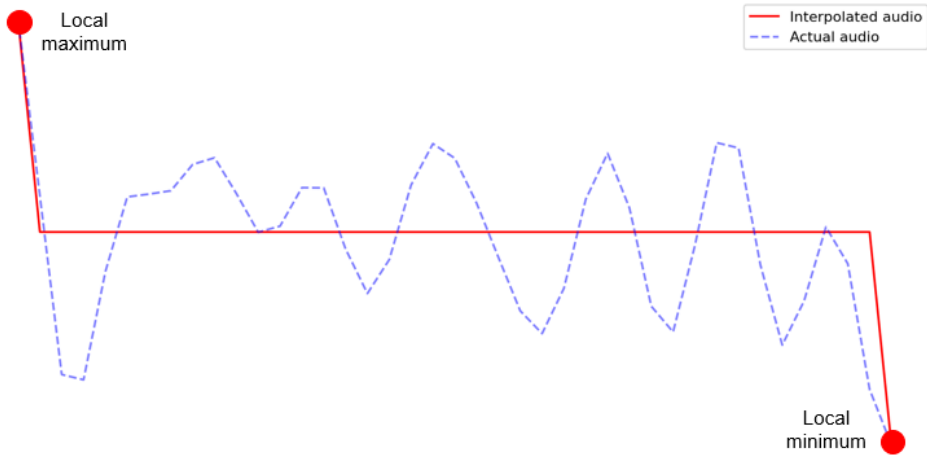
- Segments (intervals in 1D audio) obtained by identification of distinct extremes (apart enough from each other).

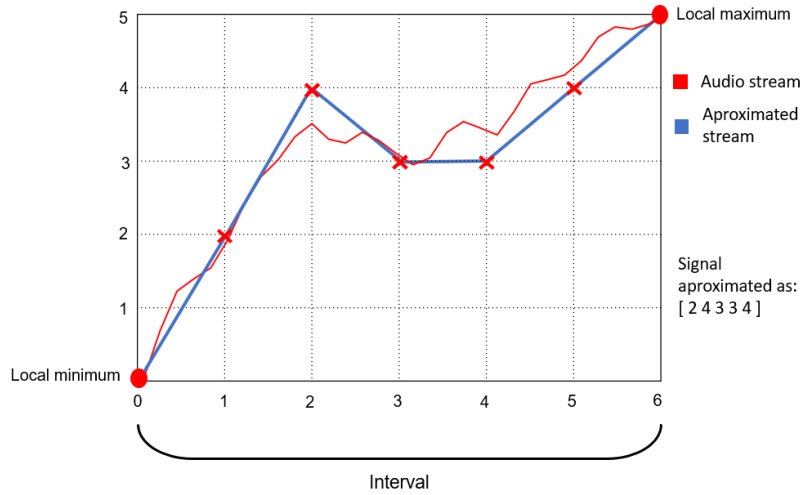


Feature-based predictions (our experience)

- Feature types:
  - Line Segment
  - Average
  - Polyline
  - RLE
  - Verbatim







Feature-based predictions (our experience)

- 7-11% worse than Monkey's audio (APE).
- Better than in early versions, but … ☹
- Feature representation = lossy compression → why not use some „good" domain-dependent lossy compression instead?
- OGG Vorbis lossy compression → only 1-2% worse than APE.
  - We even found a winning example.
- Goal: beat APE. No results today. Just concepts and discussion.

Means to achieve the goal

- OGG is not optimal in all cases.
- OGG is flexible but:
  - Low quality → small OGG file; big residuals, too much space for them.
  - High quality → bigger OGG file; small residuals, less space for them.
  - As written, 1–2% missing in both cases, with few irrelevant exceptions.
- On the other hand, segment features might interpolate data perfectly or catastrophically, depending on data.
- A combination: determine which segments to treat as segment features and which to compress with OGG.
  - Do not compress each „OGG segment“ into a separate file, but buffer the data and run a single OGG compression afterwards.

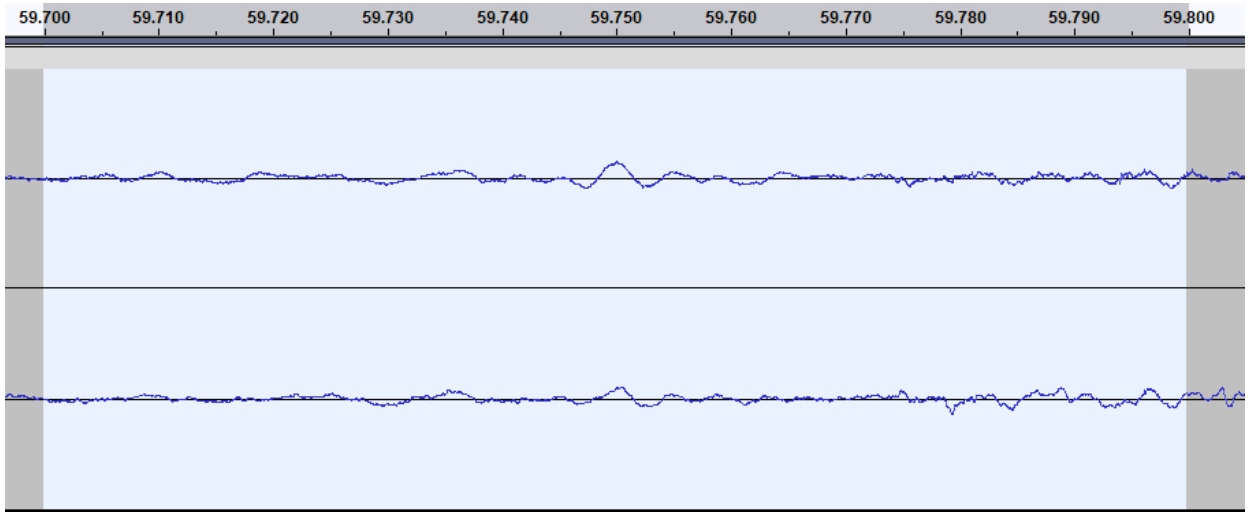
Means to achieve the goal

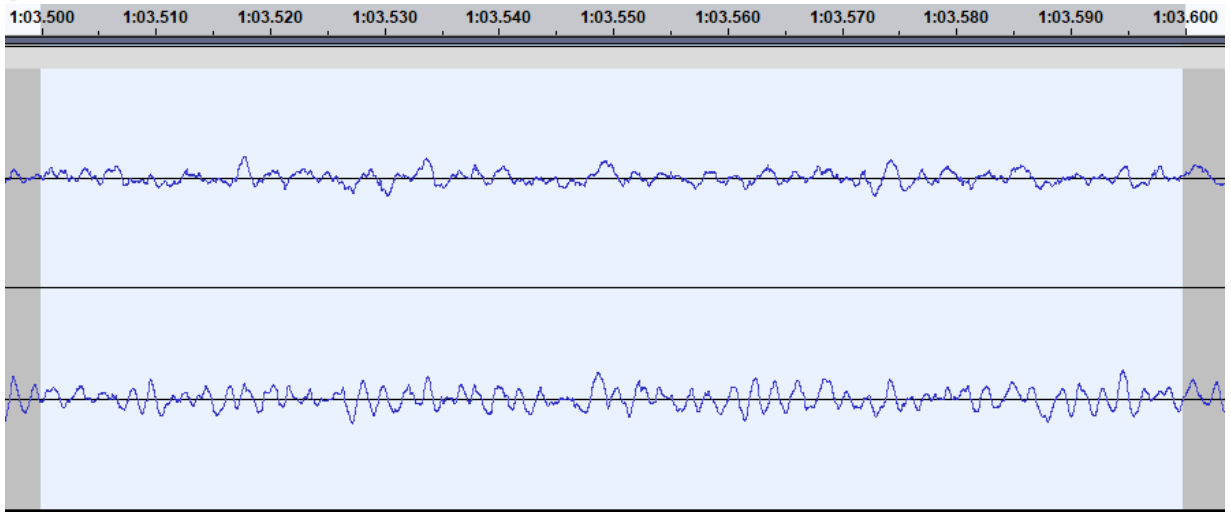
- Besides hybrid OGG-feature compression:
  - by refreshing the repertoire of feature types,
  - by a new concept of feature selection (segmentation),
  - by different entropy coding
    - Currently, residuals compressed to APE after reconstruction from OGG. Not optimal, even Rar and Zip are better with small residuals (exactly when features beat OGG!).
    - Try our implementations of BASC, AC, Deflate, Rice coding, …, with/without MTF, BWT, MwI).

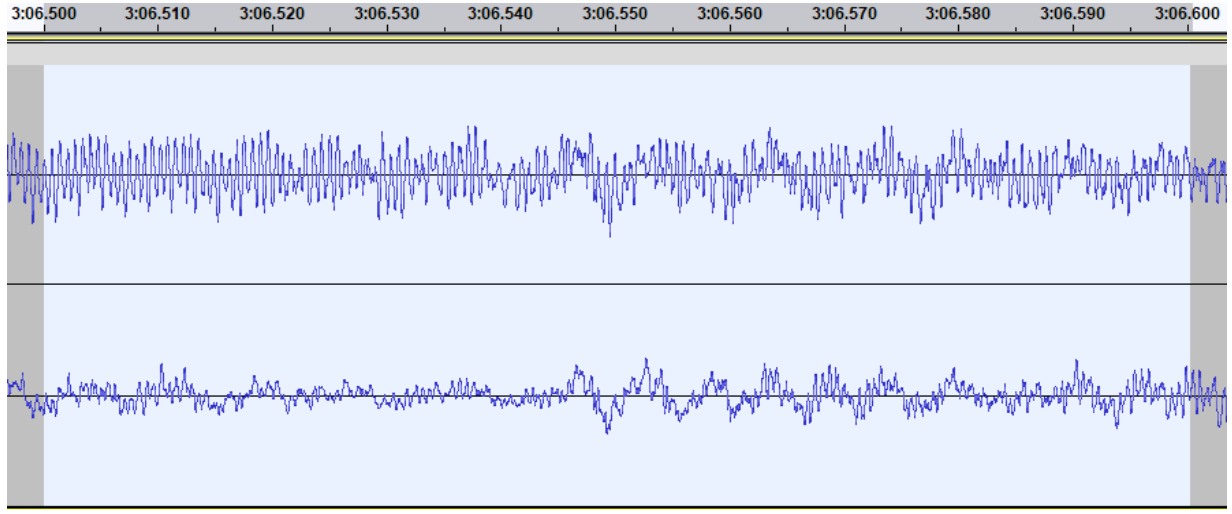
Preliminaries

- Input stream (channel) → blocks. Default length 0.1s.
- Default input: CD Audio (PCM, Stereo, 44,1 kHz, 16-bit).
- Block → intervals (segments) between pairs of successive distinct extrema (min. and max., or max. and min.).
- Optional preprocessing
  - PCM → DPCM, DDPCM (or other simple predictions).
  - Left and right channel → Central and side channel.

Blocks (4410 samples, 0.1 seconds)







Segmentation

- OGG better with high frequencies (short intervals between distinct extremes).
- New feature Restore introduced.
  - Based on parabolla through tree consecutive samples.
  - $y = a*x^2 + b*x + c$ through (x_i, y_i), (x_{i+1}, y_{i+1}), (x_{i+2}, y_{i+2}):
  - $a = (y_i - 2y_{i+1} + y_{i+2})/2$
  - $b = y_{i+1} - y_i - a \cdot (x_i + x_{i+1})$
  - $c = y_i - a \cdot {x_i}^2 - b \cdot x_i$
- It provides a unified treatment of parabolas, line segments and RLE.
- Only Polyline will be used next to OGG (for Verbatim) and Restore.

New feature Restore

- Unified treatment of parabolas, line segments and RLE.

Parabola:	$a \neq 0$
Non-horizontal line segment:	$a = 0, b \neq 0$
Horizontal line segment (RLE):	$a = b = 0$

- After determining a, b and c for a triplet of consecutive samples, the algorithm extrapolates the parabola (or line segment) until the error becomes too large.
  - Such greedy approach is not optimal and has a potential for improvements.
- Lossless, near-lossless and lossy mode in the same framework.

Usability test

- Before implementing the new feature Restore, we analyzed on 20 songs, how often parabolas, line segments and RLE are met in practice.
- Percentage of samples included in Restore intervals below 1%.
- However, this feature can be used as a good prediction, even where errors are above zero. In general, it produces low errors (residuals).
- Lossless, near-lossless and lossy mode in the same framework.
- With residual treshold 256 or 512 (9 or 10 bits for an uncompressed residual) and minimal interval length 20, about 5% samples were addressed by Restore.

New concept (to be tested soon)

1. Identify intervals Restore.
2. Identify intervals Polyline between consecutive Restore pairs.
3. Compress features and residuals with the selected method (BASC, AC, Rice, Deflate, with/without MTF, BWT, MwI).
4. Other data (blocks or segments) are buffered and stored as OGG Vorbis. Of course, residuals are stored for this segments as well.

NOTE: Instead of OGG Vorbis, ANN (autoencoder) can be used, but this requires an intensive training (perhaps in new project).