videoparser-ng

An open source, fast and scalable bitstream parser

Werner Robitza, VQEG Meeting Dec. 2025

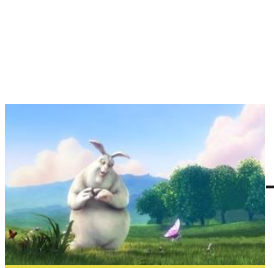
Agenda

— Video model types, an overview – and P.1204.3

— Existing bitstream parsers, and our new one

— Details and future directions

Pixel-Based Video Quality Models



Original Video

Compression

Encoded Video

Network Transmission

Played Video

Including stalling, quality switches, …

Checks for video problems

No-Reference

Full-Reference

Measures encoding quality

Via screen capture

No-Reference

Measures streaming quality

Full-Reference

Pixel-based models require large data transmission. Full-reference models can be accurate but very computationally intensive.

Bitstream- and Metadata-based Video Quality Models

Original Video

Compression

Encoded Video

Network Transmission

Played Video

Including stalling, quality switches, …

Bitstream-based model

Measures encoding quality

Via device network API or side data from network

Bitstream-based model

Metadata-based model

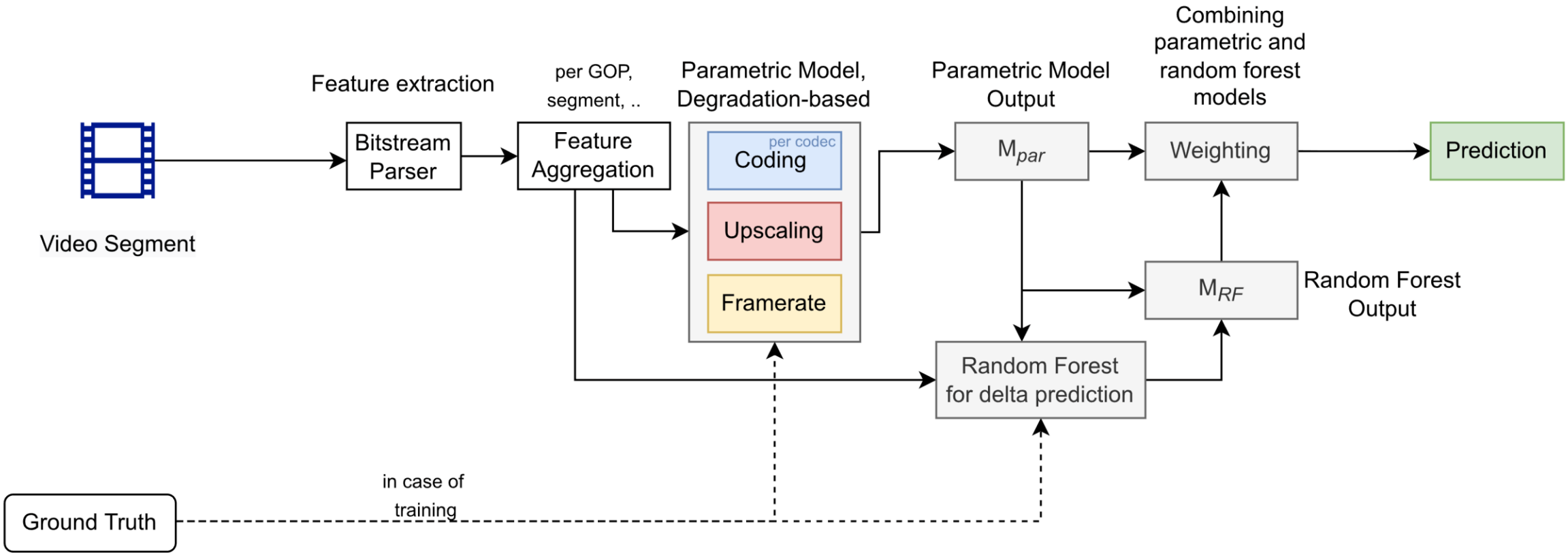
Via player API

Bitstream- or metadata-based models can be applied in different scenarios and are easy to compute.

ITU-T P.1204.3 and Bitstream Parsing

— Bitstream-based models require parsing of different kinds of video sources (containers, codecs)

— Example model that achieves great accuracy comparable to full-reference model: P.1204.3



Example from AVQBits model architecture, part of P.1204.3 bitstream model, see Rao et al. – AVQBits—Adaptive Video Quality Model Based on Bitstream Information for Various Video Applications, IEEE Access (2022)

A Quick History of Bitstream Parsers

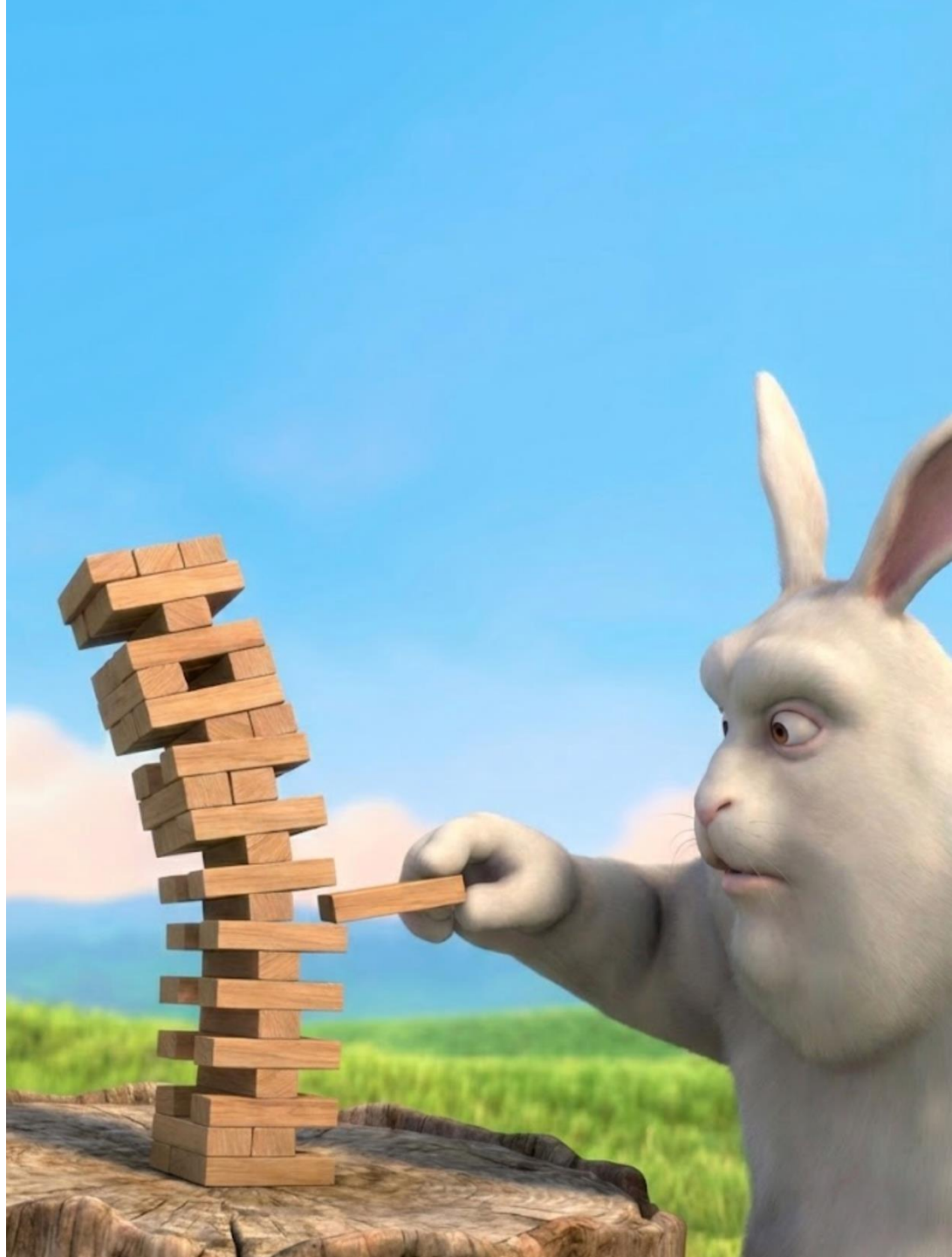
- Erana264 (2010?) – Ericsson GUI-based bitstream analyzer, defunct
- Elecard StreamEye – Commercial tool to visualize macroblocks, motion vectors, …
- VQEG JEG-Hybrid (2010–2012?) – hybrid model development project
  - Development of an H.264 bitstream data format (HMIX) by modifying H.264 encoder
  - Assumed packet loss → less practical relevance now
- VideoParser from P.1203 development
  - Based on FFmpeg and modified decoders

Legacy VideoParser

- Written by Deutsche Telekom in collaboration with TU Berlin/TU Ilmenau for model development during P.1203 phase (P.1203.1 Mode 3 → bitstream model)
- Worked well during model training and validation
- Suboptimal speed due to tradeoff between “getting as many metrics as possible” and optimization
- Code base became unmaintained and could not be compiled on newer systems
- No AV1 support

→ Long-term issues for future developments production deployments

https://github.com/Telecommunication-Telemedia-Assessment/bitstream_mode3_videoparser



The New *videoparser-ng*

- Free and open source (LGPL-licensed)
- Based on FFmpeg and libaom for extracting bitstream data
- C++ API and CLI interface
- Linux and macOS support, amd64/arm64
- Static binaries for download

Features:

- Clean, streaming JSON format as output
- H.264, HEVC, VP9, AV1
- QP parameters and motion vectors
- Extensible platform with unit tests

https://github.com/aveq-research/videoparser-ng

Bitstream Data and Speed Comparisons

Type	Metrics
QP	Avg., min, max, std. deviation
Motion vectors	Avg. MV length, std. deviation, avg. X and Y components, motion vector residuals
Macroblock stats	Blocks with MVs, explicitly coded MVs, Bit count for motion and transform coeffs
Metadata	Frame type, codec, bitrate, width/height, profile, …

Speed improvement examples:

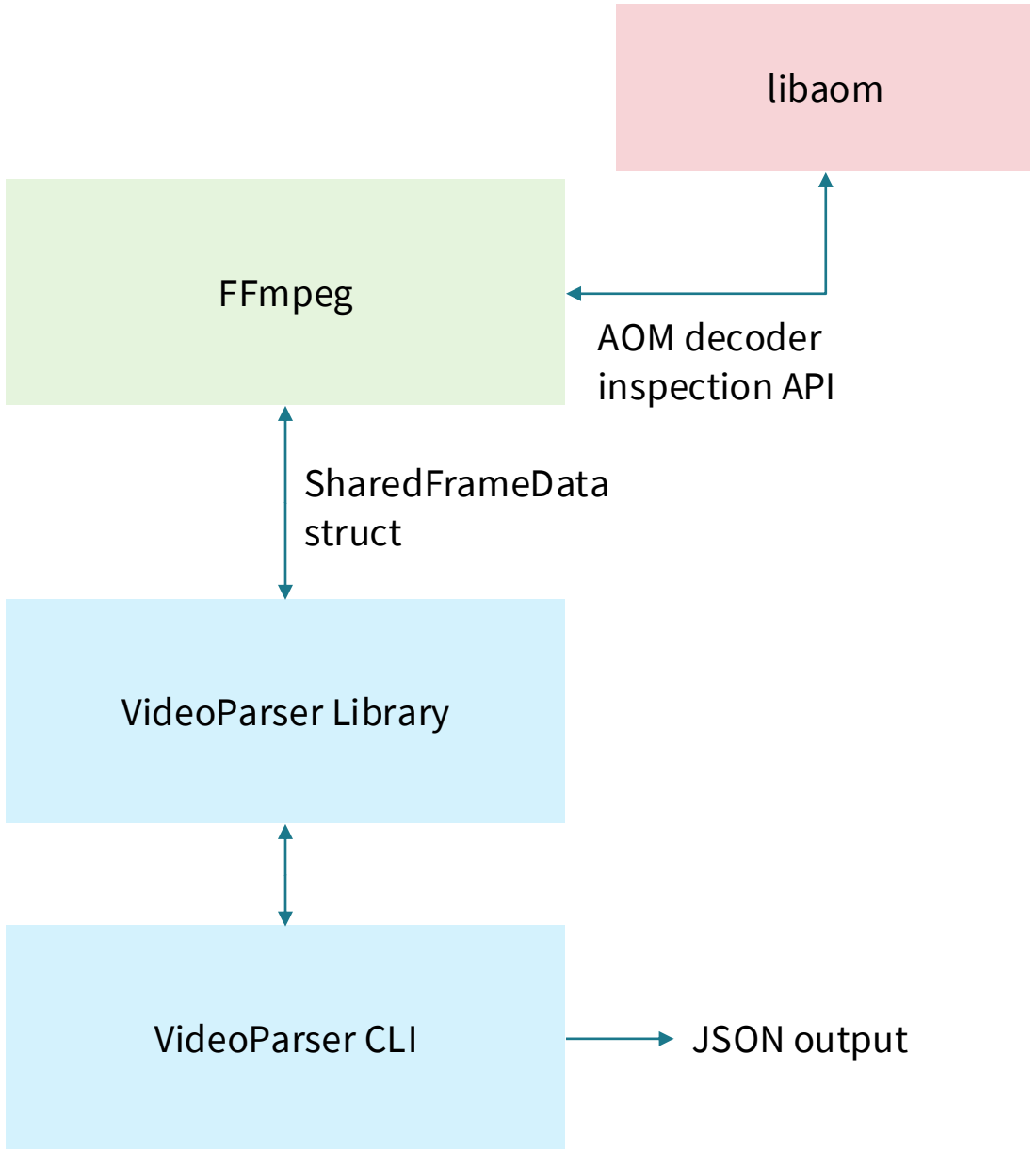
— 16x speedup for 1080p HEVC

— 70x speedup for 2160p H.264

(M1 Mac, comparing old amd64 VideoParser against arm64 videoparser-ng)

How it Works

- Modified FFmpeg (libavcodec) 8.0 to extract the data
- Modified libaom (because FFmpeg has no built-in AV1 decoder)
- Lightweight C++ library to pass data from Ffmpeg
- CLI to output data in structured format



Feature Engineering on the Raw Stats

- Raw bitstream statistics cannot be directly used for quality estimation
- Feature engineering based on per-GOP or per-segment statistics
- E.g. as in the P.1204.3 case, feature engineering done by Rakesh Rao and Steve Göring during model development/training process

→ The aggregation is up to the consumer of the raw statistics.

Aggregated Feature	Type
Framerate	float
Resolution ($width \times height$) of the distorted video	int
Codec (H.264, H.264_10bit, H.265, H.265_10bit, VP9)	boolean
$Q_{parametric}$	float
Mean bitrate per segments	float
Maximum frame size	int
Kurtosis of the non-I frame sizes	float
Standard deviation of frame size of non-I frame in bits	float
Quant ($Quant = \frac{QP_{non-Iframes}}{QP_{max}}$)	float
IQR of the average QP of non-I frames	float
IQR of the minimum QP per frame	float
Kurtosis of the average QP of non-I frames	float
Mean of the average QP of non-I frames	float
Standard deviation of maximum QP of non-I frames	float
Kurtosis of the average motion per frame over all frames in a segment	float
Minimum standard deviation of motion in the x-direction (horizontal motion) per frame	float

From: Raake et al., Multi-Model Standard for Bitstream-, Pixel-Based and Hybrid Video Quality Assessment of UHD/4K: ITU-T P.1204, IEEE Access (2020)

Future Directions

Legacy Mode for P.1204.3

— The “old” video parser had some normalizations built-in, i.e., feature engineering at the extraction level

— The new videoparser-ng does not do that, but it has a “legacy” mode to emulate it (because P.1204.3 was trained that way). Currently WIP.

Metric	New Implementation	Old Implementation	Difference
per_sequence	4.118	4.087	+0.031
baseline_mos	4.165	4.163	+0.002
rf_pred	4.071	4.011	+0.060
bitrate	6446.1	6621.5	-175.4
quant	0.6402	0.6407	-0.0005
mean_qp_non_i	32.65	32.67	-0.02

→ Used for reimplementing P.1204.3 (validation ongoing)

More Metrics

— Provide in-depth statistics e.g. on MB types, CU sizes, … based on compile-time flags

— Black border detection for QP values

Extending Codec Support

— Adding VVC support

— Retraining of P.1203.1 for AV1

Thank You!

The original VideoParser:

— **Peter List** (Deutsche Telekom) for the initial code

— **Steve Göring, Rakesh Rao Ramachandra Rao, Anton Schubert (TU Ilmenau)** for troubleshooting builds and adding features

The new videoparser-ng:

— **Jonatan Stenlund** (Ericsson, Luleå University of Technology) for implementing initial motion vector support for HEVC

— **Bob Forsman** (Ericsson) for providing bug reports and API suggestions

Contact us

Werner Robitza
CEO

werner.robitza@aveq.info
https://aveq.info