LAPSE

Language-agnostic subtitle synchronization against speech activity

Technical report

Rasmus Stisen Jensen

https://github.com/Schwponaco-org/lapse

September 2026

Versions covered.

Engine described	LAPSE 2.2.3, commit `c76f550`, 24 September 2026.
Measurements	LAPSE 2.0.0 (commit `aa975e8`, 10 August 2026) against alass 2.0.0 and ffsubsync 0.5.1, published in `docs/benchmarks.md` on 11 August 2026 (commit `3151a67`).
What differs	The speech detector, both offset searches, the lock statistic, mode selection and the verdict are the same code in 2.0.0 and 2.2.3. The later releases add subtitle formats, bitmap tracks as a reference, snapping to picture cuts, output encodings and a batch mode (Section 4.8). No number in this report was measured on 2.2.3.

Abstract

LAPSE moves a subtitle file onto the speech in a video. It corrects a constant offset, the linear drift that a framerate conversion leaves behind, and the piecewise offsets of recuts, ad breaks and files that hold two parts, and it does this without reading the language of either side. Speech is detected with Silero VAD, or libfvad when the neural model is not available, and turned into weighted spans. Offsets are found with a two channel FFT cross correlation on speech blocks and speech onsets, and both channels have a running mean removed before they are correlated. This report shows that the step is equivalent to high pass filtering the score curve, which takes out the broad hill that dense stretches of dialogue raise in a plain overlap score. Candidates are refined on an exact overlap score and ranked by a lock statistic that compares each offset with the same file at random offsets, over the whole file and in eight slices. For files that need splitting, a penalised dynamic programme [3] runs on a coarse grid, with a penalty that grows with the logarithm of the number of cues. A decision procedure picks shift, drift, recut, joined parts or restart for each file, and a verdict decides whether the original subtitle is overwritten.

On 39 feature films chosen for being hard to synchronize, LAPSE 2.0.0 passed 36 in the shift and drift test, against 31 for alass 2.0.0 and 33 for ffsubsync 0.5.1. With splitting forced on it passed 32 with 3 partial. Paired exact tests put these differences in perspective: only the gap to ffsubsync's split mode ($p = 0.0005$) is clear on this sample. On the 27 films where verdicts were recorded, all 22 answers the engine marked *solid* in the shift and drift test were right, and both of its wrong answers were held back. Automatic split detection is the weak part: it passed 19 films against 32 when splitting was forced, and its verdict does not look at the split. The report gives the algorithms with proofs where a proof exists, the libraries they rest on, the benchmark protocol in full, and the designs that were tried and dropped.

Contents

1 Introduction

1.1 The problem

A subtitle file downloaded for a film was timed against some release of that film, and rarely the one on your disk. The mismatch comes in three shapes. The whole file can be early or late by a constant, anything from a few hundred milliseconds to more than a minute. It can drift, because it was timed at 25 frames per second and the video plays at 23.976, so the error grows by a little over four seconds every hundred seconds. Or it is right for forty minutes and wrong after that, because the release it was made for has a scene the other one lacks, or the other way round.

Fixing a constant offset by hand takes a minute. Drift and recuts take much longer, and across a library of a few thousand files, none of the three is done by hand. The automatic approach suggests itself: find where people talk in the audio, and slide the subtitle until its cues sit on top of that talking. Every tool in this area does some version of that [3, 5, 10].

1.2 What makes it hard

Three things go wrong, and most of this report deals with one of them at a time.

The first is detection. A film soundtrack is a poor place to look for speech. Score, effects and crowds are loud; dialogue is sometimes whispered, sometimes shouted over an engine, sometimes on a channel the downmix buries. Any detector will call some music speech and miss some speech.

The second is the score. Even with perfect detection, the obvious measure of fit, how many cues land on speech, is highest wherever the film talks most. A film with one quiet half hour and one busy one raises a broad hill in that score, and on some films the correct offset is a narrow spike on the flank of the hill and loses to its top (Section 6.2).

The third is knowing when you are wrong. Every tool will return an offset for any input, including a subtitle for a different film. A tool that overwrites the file whether or not the answer is supported by the audio turns a problem the user can see (subtitles out of sync) into one they cannot (subtitles synced to the wrong film, with the original gone).

1.3 Approach

Both sides become sequences of time spans, and the fit of an offset is a weighted sum of normalised overlaps [3]. The search runs as an FFT cross correlation on two channels, one for where speech is and one for where it starts, with the slow trend removed from both. The best few candidates are refined on the exact overlap score and ranked by a lock statistic that asks how unusual the offset is compared with random offsets on the same file. Slices of the file are aligned on their own, and their agreement decides whether the file is shifted, drifting, recut or made of parts. A verdict built on the lock statistic and the slice agreement decides whether the result replaces the original or goes to a sidecar file beside it.

1.4 Contributions

The report makes four contributions.

1. A two channel correlation with the running mean removed, and a short proof that removing the mean from both signals is the same as high pass filtering the score curve with the autocorrelation of the filter (Proposition 6.6). The filter removes the slow component of the score, which is where the hill lives.
2. A lock statistic that compares an offset against the file's own score at random offsets, both globally and in eight slices, with bounds and invariance properties (Section 7).
3. A procedure that picks the kind of correction per file, with a split penalty scaled to the file size (Section 10).
4. A verdict that decides whether the original is overwritten, and a record of how well it matched the outcomes on a hard benchmark (Section 12.10).

None of these is deep on its own. Together they give an engine that passes more hard films than the tools it was compared with and, more usefully, flags most of the films it gets wrong.

The name owes something to the title of Kaegi's thesis, *Language-Agnostic Subtitle Synchronization*, and so does a good part of the engine.

1.5 Versions and scope

The report describes the engine as it stands in version 2.2.3. All measurements come from a benchmark of version 2.0.0, published on 11 August 2026. Between the two releases the speech detector, the searches, the lock statistic, mode selection and the verdict did not change; Section 4.8 lists what did. Where this report quotes a number, it is a 2.0.0 number.

The Docker watcher, the web interface and the translation feature that ship with LAPSE are left out. So are the subtitle parsers and writers, except where they affect timing.

1.6 Structure

Section 2 places LAPSE among existing tools and methods, and Section 3 sets up the notation and the objectives. Section 4 covers the pipeline and the libraries. Sections 5 to 10 follow the pipeline from speech detection to the verdict, each ending with a short account of the designs it replaced, and Section 11 follows one film through all of it with the numbers the engine computed. Section 12 describes the benchmark and its results, Section 13 the development timeline, and Section 14 the threats to validity and the limits of the engine.

2 Background and related work

2.1 How subtitles go out of sync

Kaegi [3] sorts the errors found in subtitles from online databases into three patterns, and LAPSE uses the same list. A *shift* is a constant offset, typically from a release with a different intro or a different lead in before the first frame. A *framerate difference* scales the time axis, most often between 23.976, 24 and 25 frames per second, because PAL releases speed a 24 fps film up to 25. *Splits* are points where the offset jumps, from an added or removed scene, an ad break in a television capture, or two discs joined into one file. A single file can show more than one pattern: a PAL release of a director's cut both drifts and jumps.

2.2 Span alignment: alass

Kaegi's thesis and its implementation, alass, turn the problem into aligning two sequences of time intervals. The reference sequence comes from a voice activity detector run on the film, or from a second subtitle that is known to be right, and the input sequence comes from the subtitle to be fixed. The score of an alignment is a sum over pairs of spans of their overlap, normalised by the shorter span and weighted by the ratio of lengths, minus a penalty for every split. Two algorithms matter here. The first finds the best single offset exactly: the score of one pair is a trapezoid in the offset, so the total is piecewise linear with at most $4KN$ breakpoints, and a sweep over the sorted breakpoints finds the maximum (his Algorithm 4, with counting sort in Algorithm 5). The second finds the best split alignment by dynamic programming over offsets, with a constraint that keeps cues in order (his Algorithms 6 and 7). For framerates, alass scales the input by each of seven ratios and keeps the one with the highest score.

On more than 20 films and 100 subtitle files, the thesis reports an error rate of 12% for alignment against the film and none for alignment against another subtitle. The reference implementation uses the WebRTC voice activity detector and splits by default, at a penalty of 7. LAPSE takes the span representation, the pair score, the sweep and the split recursion from this work.

2.3 Binary strings and FFTs: ffsubsync

ffsubsync [5] discretises the audio and the subtitle into 10 ms windows, marks each window as speech or not, and aligns the two binary strings by maximising their agreement, computed for all offsets at

once with an FFT. The default detector is the WebRTC VAD. A small set of framerate ratios is tried. An alass style piecewise mode is available behind `--split-penalty`, described as experimental and off by default, and a score is printed that can be used to skip writing a poor result, also off by default. The benchmark in Section 12 uses version 0.5.1.

2.4 Recognition and learned detectors

A second family of tools recognises words. subsync [13] runs speech recognition on the audio and matches the recognised words with the words in the subtitle, which gives precise anchors when it works, at the price of needing a language model for the language of the film and of the subtitle. autosubsync [10] stays language agnostic but trains a small classifier to detect speech in the audio and searches jointly over offset and framerate. LAPSE does not recognise words. It aims to work on any pair of languages, including a subtitle in one language against a film in another, and to run in seconds on a machine without a GPU.

2.5 Voice activity detection

Voice activity detectors come in roughly three generations. Energy based detectors threshold the short term energy of the signal, sometimes restricted to the telephone band of 300 to 3 400 Hz where most of the energy of speech sits. They are cheap and easily fooled by anything loud. The WebRTC detector, available as a library in libfvad [6], computes log energies in six sub bands and feeds them to Gaussian mixture models for speech and noise, with four modes of increasing aggressiveness; it makes a decision every 10, 20 or 30 ms. Neural detectors such as Silero VAD [12] run a small recurrent network on raw audio in windows of 32 ms and return a probability. They are much better at telling speech from music, and still fast enough to run over a two hour film in seconds on one core. LAPSE uses Silero and falls back to libfvad.

2.6 Changepoints and warping

Split alignment is a changepoint problem. The offset is piecewise constant along the file, and each change is paid for with a penalty. Penalised optimal partitioning [2] solves this kind of objective exactly by dynamic programming, and PELT [4] prunes the search to make it linear in the typical case. The penalty that grows with $\ln n$ has the form of the Bayesian information criterion [9]. Dynamic time warping [8] is the familiar tool for aligning two time series, but it allows the alignment to change at every step. Subtitles need the opposite: a recut film jumps and then holds still.

2.7 Median based statistics

Several parts of LAPSE need a location and a scale that ignore outliers: the noise floor of a correlation curve, the offset shared by most slices of a file, the slope of a drifting file. The engine uses the median, the median absolute deviation scaled by 1.4826 so that it estimates the standard deviation under a normal model [7], and the Theil and Sen estimator of a slope, the median of all pairwise slopes [11, 14], which tolerates corruption of up to about 29% of the points.

3 Formal model

This section fixes the notation used for the rest of the report. The span representation, the pair score, the span preparation of Algorithm 1, the exact sweep of Algorithm 4 and the split recursion of Theorem 9.3 follow Kaegi [3]; the weights h_k, the trend removed search, the lock statistic, the mode selection and the verdict are LAPSE's own.

3.1 Spans and sequences

Definition 3.1 (Span)**.** A *span* is a half open interval $a = [a_1, a_2)$ with $a_1 < a_2$, in integer milliseconds. Its length is $\mathrm{len}(a) = a_2 - a_1$, and for $\sigma \in \mathbb{R}$ the shifted span is $a + \sigma = [a_1 + \sigma, a_2 + \sigma)$.

Definition 3.2 (Valid sequence)**.** A sequence of spans $(a^1, \dots, a^N)$ is *valid* if $a_2^n \le a_1^{n+1}$ for every n: the spans are ordered and do not overlap.

The subtitle yields the *input* sequence $c = (c^1, \dots, c^N)$, one span per cue after overlapping cues are merged (Section 4.5). The film yields the *reference* sequence $r = (r^1, \dots, r^K)$ with weights $h_k \in (0, 1]$. When the reference comes from a VAD, h_k is the mean speech probability inside the span; when it comes from another subtitle or an embedded track, $h_k = 1$. Both sequences are valid.

3.2 Error models

Write t for a time in the subtitle and t' for the time the same line is spoken in the video. The three patterns of Section 2 become three models:

$$\text{shift:} \quad t' = t + \sigma, \tag{1}$$

$$\text{drift:} \quad t' = \rho t + \sigma, \qquad \rho \in P, \tag{2}$$

$$\text{split:} \quad t' = t + \sigma_n \quad \text{for cue } n, \qquad \sigma_n \text{ piecewise constant in } n. \tag{3}$$

P is a finite set of framerate ratios (Table 2), and the drift model can be combined with the split model. LAPSE reports ρ as `ratio` and σ as `offset_ms`.

3.3 The pair score

Definition 3.3 (Overlap and pair score)**.** For spans a, b,

$$\operatorname{ov}(a, b) = \max\big(0,\ \min(a_2, b_2) - \max(a_1, b_1)\big), \qquad \operatorname{iscore}(a, b) = \frac{\operatorname{ov}(a, b)}{\min(\operatorname{len} a, \operatorname{len} b)}, \tag{4}$$

and the pair weight is $w(a, b) = \min(\operatorname{len} a, \operatorname{len} b) / \max(\operatorname{len} a, \operatorname{len} b)$. The *pair score* of reference span r^k and cue c^n at offset σ is

$$\phi_{kn}(\sigma) = h_k \, w(r^k, c^n) \, \operatorname{iscore}(r^k, c^n + \sigma). \tag{5}$$

The normalisation makes a perfect fit worth h_k only when the two spans have the same length. A two second cue sitting inside a ten second speech span covers itself completely, so iscore $= 1$, but $w = 0.2$, so the pair is worth a fifth of a perfect match. The product has a simpler form.

Lemma 3.4. $\phi_{kn}(\sigma) = h_k \, \operatorname{ov}(r^k, c^n + \sigma) \,/\, \max(\operatorname{len} r^k, \operatorname{len} c^n)$.

Proof. $w \cdot \operatorname{iscore} = \dfrac{\min(\operatorname{len} r, \operatorname{len} c)}{\max(\operatorname{len} r, \operatorname{len} c)} \cdot \dfrac{\operatorname{ov}}{\min(\operatorname{len} r, \operatorname{len} c)} = \dfrac{\operatorname{ov}}{\max(\operatorname{len} r, \operatorname{len} c)}$. Shifting a span does not change its length. □

Lemma 3.5 (Trapezoid)**.** *Let $\ell = \min(\operatorname{len} r, \operatorname{len} c)$ and $L = \max(\operatorname{len} r, \operatorname{len} c)$. As a function of σ, $\phi(\sigma)$ is continuous and piecewise linear with four breakpoints*

$$\beta_1 = r_1 - c_2, \quad \beta_2 = \beta_1 + \ell, \quad \beta_3 = r_2 - c_1 - \ell, \quad \beta_4 = r_2 - c_1.$$

It is zero outside (β_1, β_4), rises with slope h/L on $[\beta_1, \beta_2]$, is constant at $h\ell/L$ on $[\beta_2, \beta_3]$, and falls with slope $-h/L$ on $[\beta_3, \beta_4]$. All four breakpoints are integers.

Proof. The shifted cue $c + \sigma$ meets r exactly when $c_2 + \sigma > r_1$ and $c_1 + \sigma < r_2$, that is on (β_1, β_4). Inside that interval $\operatorname{ov} = \min(r_2, c_2 + \sigma) - \max(r_1, c_1 + \sigma)$, whose derivative in σ is $\mathbf{1}[c_2 + \sigma < r_2] - \mathbf{1}[c_1 + \sigma > r_1]$. For σ just above β_1 the cue's end has entered r and its start has not, so the derivative is $+1$. It stays $+1$ until either the start of the cue reaches r_1 (when $\operatorname{len} c \le \operatorname{len} r$) or the end of the cue reaches r_2 (when $\operatorname{len} c > \operatorname{len} r$). Both happen at $\sigma = \beta_1 + \ell$. Then the shorter span lies inside the longer one and the derivative is 0 until the start of the shorter one would leave the longer one, at $\beta_4 - \ell = \beta_3$. From there to β_4 the derivative is -1. Lemma 3.4 scales all of this by h/L. The breakpoints are sums and differences of integer timestamps. □

The four breakpoints and the slope changes at them, $+h/L$, $-h/L$, $-h/L$, $+h/L$, are what the code calls `sig1` to `sig4`. The plateau starts at $r_1 - c_1$ when the cue is the shorter span and at $r_2 - c_2$ when the reference span is; both equal $\beta_1 + \ell$.

3.4 Objectives

The single offset score is

$$S(\sigma) = \sum_{n=1}^{N} \sum_{k=1}^{K} \phi_{kn}(\sigma). \tag{6}$$

A cue that lands exactly on a span of its own length contributes h_k, so $S(\sigma)/N$ reads roughly as the share of the subtitle that sits on speech. The engine reports $\min\bigl(1, S(\hat{\sigma})/N\bigr)$ as `confidence`. For the drift model the cues are scaled first, $c^n \mapsto \rho c^n$, and $S_\rho(\sigma)$ denotes the score of the scaled sequence. For the split model every cue has its own offset and

$$S(\sigma_1, \dots, \sigma_N) = \sum_{n} \sum_{k} \phi_{kn}(\sigma_n) \; - \; p \cdot \#\{n : \sigma_n \neq \sigma_{n-1}\}, \tag{7}$$

maximised subject to the shifted sequence $(c^n + \sigma_n)$ staying valid. Since no cue is worth more than 1, the penalty p reads as a number of cues: a split is taken only if it gains the equivalent of more than p perfectly placed cues.

3.5 How many cues are enough

With very few cues the objective cannot tell offsets apart. A rough model makes the point. Suppose the reference is speech a fraction d of the time, and a random offset puts each cue on speech independently with probability d. Then all N cues land on speech at a random offset with probability about d^N. For $d = 0.4$ and $N = 4$ that is 2.6%, and a search over ±15 minutes in 40 ms bins looks at 45 000 offsets, so more than a thousand of them fit perfectly. For $N = 10$ it is 10^{-4}, or a handful of offsets. The independence assumption is optimistic, since cues cluster, but the conclusion stands, and the engine refuses files with fewer than 10 cues unless `--force` is given. The case is common: many release groups ship a subtitle file that holds nothing but their own name.

4 Architecture and libraries

4.1 The pipeline

Figure 1 follows one run of the engine. The subtitle branch and the reference branch meet in the offset search, and everything after that point works on spans only.

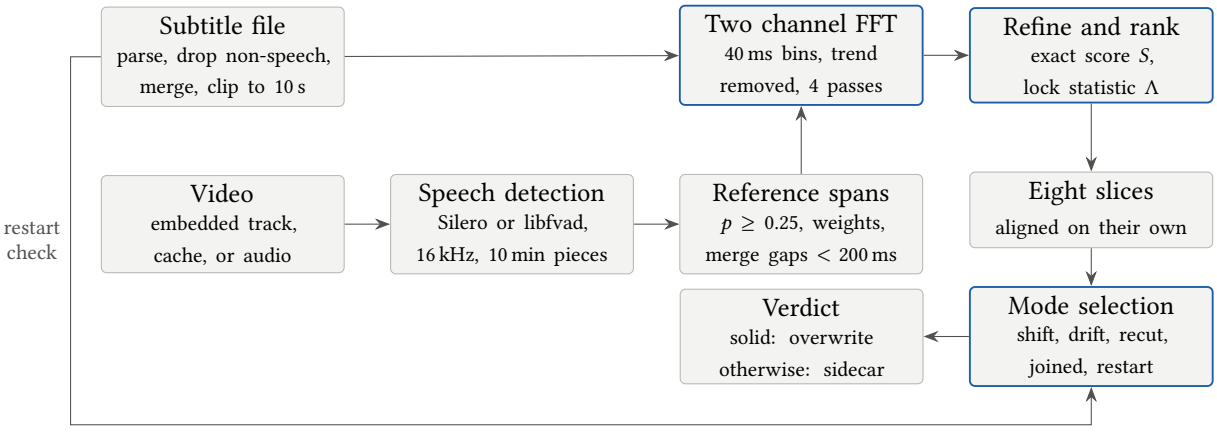

Figure 1. One run of the engine. Boxes with a blue frame are covered in Sections 6, 7 and 10. A reference subtitle file skips the audio branch, and a file that restarts its timestamps goes straight to mode selection.

4.2 Where the reference comes from

The reference is taken from the cheapest source that has one, in this order:

1. another subtitle file, when the first argument is one;

2. a subtitle track inside the video. Text tracks give their cue times directly. PGS and VobSub tracks are bitmaps with no text, but their packet times are exact, so a span runs from one packet to the next, capped at 10 s;
3. speech spans cached from an earlier run on the same video (Section 4.7);
4. speech detection on the audio (Section 5).

A subtitle reference is exact and needs no decoding; audio is the slowest and noisiest source by a wide margin. In a library the cache covers the common case, where one film has several subtitle tracks and each needs syncing. In the benchmark a cached film took a median of 0.66 s against 10.82 s for the first run.

4.3 Libraries

LAPSE is a C++ program of about 6 000 lines, built on four libraries. Table 1 lists them with the role each plays and what was weighed against it.

Library	Role in LAPSE	Alternatives and why not
FFmpeg	Demuxing every container, decoding audio, resampling to 16 kHz mono with a chosen mix matrix, reading keyframes and embedded subtitle packets.	None realistic for this breadth of formats. GStreamer covers similar ground with a much heavier runtime.
FFTW3 [1]	Real to complex and complex to real transforms of length 2^k for every correlation.	pocketfft and KissFFT are smaller and permissively licensed; FFTW is usually faster, and its GPL licence matches LAPSE's own.
ONNX Runtime	Running the Silero model. Loaded at run time, so the binary starts without it.	Loading it at run time lets the binary start, and fall back to libfvad, on a machine where the runtime is missing or broken.
Silero VAD [12]	Speech probability per 32 ms window.	WebRTC (below) is weaker on music. Larger neural detectors, such as the segmentation models in pyannote, need PyTorch, a heavy dependency for a single binary.
libfvad [6]	Fallback detector when ONNX Runtime or the model is missing.	It is the WebRTC detector with a C interface and no other dependencies.

Table 1. Libraries and their roles.

FFmpeg. Audio is decoded with libavcodec and passed through libswresample, which does the channel mix and the resampling to 16 kHz in one step. The mix is chosen per file (Section 5.2): either the default downmix or an explicit matrix that keeps only the centre channel or averages front left and right. Seeking uses `AVSEEK_FLAG_BACKWARD` and a priming margin of 3 s, so a piece decoded from the middle of the film starts on a clean frame. Picture cuts for snapping come from the packet index with `AVDISCARD_NONKEY` set on the video stream, so no frame is decoded.

FFTW. Plans are made with `FFTW_ESTIMATE`, which picks an algorithm without timing candidates. With `FFTW_MEASURE`, planning would cost more than the transforms themselves in a program that plans once per run. The reference transforms are computed once per film and reused for every query, and queries execute the same plan on new arrays through the new array interface (`fftw_execute_dft_r2c`). The first versions saved FFTW wisdom to a file; the engine now keeps no FFT state between runs.

ONNX Runtime. The library is found at run time with `dlopen` or `LoadLibrary`: first from `LAPSE_ONNXRUNTIME`, then beside the binary, then on the system search path. The C API is requested at the version the engine was built against and then at each older version down to 11, so a system copy that is a few releases old still works. The session runs with one intra op and one inter op thread, because the engine parallelises over pieces of the film instead (Section 5.1), and with all graph optimisations on. If any step fails, the engine says why and continues with libfvad.

One binary. Release builds link FFmpeg, FFTW and libfvad statically, so an archive holds the binary, the ONNX Runtime library, the model and the licence, and nothing else has to be installed. The same code is compiled to WebAssembly for the browser version. The Docker image wraps the binary with a file watcher and does not change what the engine computes.

4.4 Determinism

Proposition 4.1. *For a given input file and build, the speech profile does not depend on the number of threads.*

Proof. The film is cut into pieces of exactly 600 s, a constant, not a function of the core count. Each piece is decoded with its own decoder and resampler from a fixed seek point, and analysed by a detector whose state starts from zero at the start of the piece. The result is written to the positions of the profile that belong to that piece. Threads only decide which piece is worked on when; no state is shared between pieces, so the order of completion does not reach the result. □

The proposition covers threads, not machines. ONNX Runtime and FFTW choose SIMD code paths by CPU, and floating point sums in a different order can differ in the last bits, which can move a threshold decision on a frame. The lock statistic draws its random offsets from generators with fixed seeds, so it adds no variation of its own.

4.5 Subtitle preprocessing

From file to cues. Twelve file extensions are read; each parser returns a list of (start, end) pairs in file order, and a mapping records which pair came from which cue so that the writer can shift the original lines and leave everything else untouched. MicroDVD counts frames instead of time. Its rate is taken from the video, then from a `{1}{1}` header, and when the reference is another subtitle it is inferred from the running times. Parsed at a provisional 25 fps, a file whose true rate is f runs $25/f$ times as long as it should, so

$$\hat{f} = 25 \cdot \frac{T_{\text{ours}}}{T_{\text{theirs}}}, \tag{8}$$

snapped to the nearest of 23.976, 24, 25, 29.97 and 30 and rejected if it misses by more than 8%.

Cues that are not speech. Sound descriptions and song lyrics appear in the subtitle and never in the speech detector's output, so they cannot help and can hurt. A cue is excluded from scoring, while still being shifted and written back out, when every line in it is one of: empty after markup is removed; made only of music marks (♪, ♫, #, *, ~); enclosed in brackets of any of six kinds, including full width forms; opened by a bracket that never closes; or an upper case speaker label with nothing after the colon. If fewer than 10 cues would remain, the filter is switched off, since a file of lyrics is better aligned on lyrics than not at all.

Span preparation. Algorithm 1 turns the cues into a valid sequence. Cues that overlap are merged, because two lines on screen at once are one moment in the audio and counting them twice only skews the score. Each span is then capped at 10 s for scoring. A sign left on screen for forty seconds used to outweigh a dozen spoken lines just by being long.

Algorithm 1. Span preparation

Input: cue times $x^1, \dots, x^M$ in file order; flags *merge*, *sort*
Output: valid spans $c^1, \dots, c^N$ and a map $\mu : \{1..M\} \to \{1..N\}$

$Y \leftarrow \{(x^i, i) : \operatorname{len} x^i > 0\}$, swapping start and end where reversed
if *sort* **then** sort Y by start time
end if
for $(y, i) \in Y$ in order **do**
    if *merge* and c is not empty and $y_1 < c_2^{\text{last}}$ **then**
        $c_2^{\text{last}} \leftarrow \max(c_2^{\text{last}}, y_2)$ ▷ overlap: extend the last span
    else
        append y to c
    end if
    $\mu(i) \leftarrow$ index of the last span
end for
cap every span at 10 s of length ▷ scoring only, the file keeps its times
give each dropped cue the span of the cue before it

In split mode the cues are neither merged nor sorted. When a file has been cut and the two halves given different offsets, they overlap once sorted, and merging across the overlap would weld cues from both sides into a span that can never come apart again.

4.6 Writing the result

Every correction the engine produces has the same form: a slope ε, which is $\rho - 1$ and zero unless the file drifts, and an offset σ_m per span. Cue i of the original file, with the map μ of Algorithm 1, is moved to

$$t' = \max\bigl(0,\ \lfloor t\,(1 + \varepsilon) + \sigma_{\mu(i)} \rfloor\bigr) \tag{9}$$

for both its start and its end. Cues that were dropped from scoring, as sound descriptions or duplicates, follow the span of the cue before them, so they move with their neighbours. The writer walks the original file from top to bottom and rewrites only the timestamps. Text, styling, cue numbers, headers and the character encoding stay as they were, unless an encoding is asked for with `--encoding`. Each format is written at its own resolution: SubRip and WebVTT in milliseconds, SubViewer in hundredths, MPL2 in tenths of a second, and MicroDVD in frames at the rate it was read with. When the original is overwritten, a copy is first kept as `.bak`, and an existing `.bak` is never replaced.

4.7 The speech cache

Speech spans are written to a cache file named by a 64 bit FNV-1a hash of the video's path, size, modification time and the audio track number. The file holds a format tag, the share of the film that was analysed, and one line per span with its start, end and weight. It is written to a temporary name and renamed into place, so an interrupted run never leaves a half written cache. A cache with fewer than 20 spans is ignored.

4.8 Changes between 2.0.0 and 2.2.3

The files that hold the searches, `correlate.cpp` and `align.cpp`, are unchanged. `silero.cpp` gained one error message. The decision procedure and the verdict in `main.cpp` are unchanged. What was added is:

- readers and writers for MicroDVD, PGS, SubViewer 2, MPL2, SBV, VobSub, SAMI and TTML, and a check that opens files named `.txt` to see what they hold;
- embedded PGS and VobSub tracks as a reference (previously only text tracks were used);
- a more careful frame rate probe that times packets when the container's rate is missing or implausible;
- snapping cue starts to picture cuts (Section 10.5), off by default;

- a choice of output encoding, and a batch mode that keeps the Silero session loaded across files.

None of these touches the path the benchmark measured, with one possible exception: a film whose only embedded track is a bitmap track would have been synced against the audio in 2.0.0 and against that track in 2.2.3. The four `.mkv` films in the benchmark had embedded subtitles; Section 14 returns to them.

5 Speech detection

The detector turns a soundtrack into a probability of speech for every 10 ms frame, and from there into weighted spans. Nothing downstream can recover from a detector that misses the dialogue, so the section spends some time on the choices that decide what the detector hears.

5.1 Decoding

Audio is decoded to 16 kHz mono signed 16 bit samples, the rate Silero was trained at. The film is cut into pieces of 600 s, decoded on up to eight threads, each piece seeking to 3 s before its start and discarding the samples before it. With T the running time in seconds, there are $\lceil T/600 \rceil$ pieces, twelve for a two hour film. A piece that comes back empty from Silero is decoded again and passed to libfvad.

5.2 Choosing the channel

On a 5.1 track, dialogue sits almost entirely in the centre channel, and a plain downmix buries it under effects and score. Some releases have a silent or badly mastered centre, though, so the engine cannot just take it. When the track has more than two channels, up to three mixes are tried on a probe of 180 s starting at 35% of the running time: the centre channel alone, the default downmix, and the average of front left and right. Each probe is run through the detector, and the mix with the highest

$$Q = \underbrace{\frac{1}{F}\sum_{f=1}^{F} \max(p_f,\, 1 - p_f)}_{\text{decisiveness}} \cdot \pi(\bar{p}), \qquad \pi(\bar{p}) = \begin{cases} \bar{p}/0.15 & \bar{p} < 0.15, \\ 1 & 0.15 \le \bar{p} \le 0.65, \\ (1-\bar{p})/0.35 & \bar{p} > 0.65, \end{cases} \tag{10}$$

is used for the whole film, where p_f is the speech probability of frame f and $\bar{p}$ its mean. Decisiveness is $1/2$ for a detector that answers 0.5 everywhere and 1 for one that only answers 0 or 1. The plausibility factor π penalises mixes where speech would take up less than 15% or more than 65% of the time. A feature film sits comfortably inside that band, and a mix outside it is usually one where the detector is reacting to music.

5.3 Silero

Silero VAD takes windows of 512 samples, 32 ms at 16 kHz, together with a recurrent state of shape $2 \times B \times 128$ for a batch of B streams and the sample rate, and returns a speech probability per stream and an updated state. Two details of the interface matter in practice.

Context. The model expects the last 64 samples of the previous window glued to the front of the current one, so each call gets $64 + 512$ samples. Without the context the probabilities come back near zero on speech that is plainly audible.

Level. Quiet transfers are common, and Silero reacts to level. Each piece is scaled by

$$g = \operatorname{clip}\Big(\frac{0.1}{L_{90}},\ 1,\ 12\Big), \tag{11}$$

where L_{90} is the 90th percentile of the RMS level over 10 ms frames, and clipped to $[-1, 1]$. The mean would be the wrong statistic here: a film that is quiet most of the time pulls it towards zero and would be boosted into clipping in its loud scenes. The 90th percentile tracks how loud the film gets when something happens.

Lanes. The per call overhead of the runtime is larger than the arithmetic for one window. Since the model does not care how many streams it is given, a piece is cut into $B = 8$ contiguous lanes that advance together, one window of each per call, as in Algorithm 2. A piece of 600 s has 18 750 windows, so a call processes eight of them and a piece needs 2 344 calls instead of 18 750. Each lane starts with a zero state, so the model warms up eight times per piece instead of once, at fixed positions. Some exports of the model fix the batch dimension to 1; the engine detects that on a probe call and falls back to one lane.

Algorithm 2. Batched Silero over one piece

Input: samples x of one piece at 16 kHz, gain g, lane count B
Output: probability per 10 ms frame

$W \leftarrow \lfloor |x|/512 \rfloor$; **if** $W < 512$ **then** $B \leftarrow 1$
$P \leftarrow \lceil W/B \rceil$; state $\leftarrow 0$; context $\leftarrow 0$ ▷ per lane
for $t = 0, \dots, P-1$ **do**
  for $l = 0, \dots, B-1$ **do**
    $w \leftarrow lP + t$; $\text{row}_l \leftarrow \text{clip}(g \cdot x[512w \,..\, 512w + 511], -1, 1)$, or zeros if $w \geq W$
  end for
  $(q, \text{state}) \leftarrow \textsc{Silero}([\text{context}_l \,\|\, \text{row}_l]_{l<B},\ \text{state},\ 16000)$
  $\text{context}_l \leftarrow$ last 64 samples of $[\text{context}_l \| \text{row}_l]$; $\hat{p}[lP + t] \leftarrow q_l$
end for
return $p_f = \hat{p}\big[\lfloor (10f + 5)/32 \rfloor\big]$ for every frame f ▷ 32 ms windows to 10 ms frames

5.4 The libfvad fallback

libfvad makes a binary decision per 10 ms frame in one of four modes, from least to most aggressive in rejecting non-speech. LAPSE runs all four on the same frame and uses the share that say speech as a probability, $p_f \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$. Before that, a slow gain evens out the level:

$$L_f = 0.995\, L_{f-1} + 0.005\, \ell_f, \qquad g_f = \text{clip}(0.05/L_f,\ 0.5,\ 40), \tag{12}$$

where ℓ_f is the RMS of frame f. With a step of 10 ms this is an exponential average with a time constant of $10\,\text{ms}/0.005 = 2$ s.

WebRTC's detector is easily fooled by music, so a filter damps frames whose loudness hardly moves. Over the 101 frames centred on f (about one second), let CV_f be the coefficient of variation of the RMS level. Then

$$p_f \leftarrow p_f \cdot \min\big(1,\ \text{CV}_f/0.35\big). \tag{13}$$

Speech is modulated at the syllable rate, a few times per second, and its level varies strongly within a second. Sustained score and held notes vary much less. The filter is crude and has no effect on percussive music, but it removes most of what the WebRTC detector mistakes for dialogue in an orchestral score. Silero output is not filtered, since the model already separates the two better than the filter could.

5.5 From probabilities to spans

Algorithm 3 thresholds the probabilities at 0.25, closes gaps shorter than 200 ms, and drops spans shorter than 200 ms. The threshold is deliberately low. A span the detector was lukewarm about still gets in, with a low weight h_k, and the scoring decides how much it counts.

Algorithm 3. Reference spans from frame probabilities

Input: probabilities $p_1, \dots, p_F$ per 10 ms frame
Output: spans r^k with weights h_k

raw $\leftarrow$ maximal runs of frames with $p_f \geq 0.25$, each with $h =$ mean of p_f over the run
for each raw span u in order **do**
  if r is not empty and $u_1 - r_2^{\text{last}} < 200$ ms **then**
    $h_{\text{last}} \leftarrow \dfrac{h_{\text{last}} \operatorname{len} r^{\text{last}} + h_u \operatorname{len} u}{\operatorname{len} r^{\text{last}} + \operatorname{len} u}$; $r_2^{\text{last}} \leftarrow u_2$ ▷ breath inside a sentence
  else

```
        append u to r
    end if
end for
drop every r^k with len r^k < 200 ms                ▷ a door, a cough, a twitch
```

The merge is length weighted, so the weight of a merged span is the mean probability over its speech frames, and the gap itself does not count. Both 200 ms constants have the same reading: nobody says anything in a fifth of a second, and nobody breathes in less.

> **Design history.** The first version of LAPSE, on 3 July 2026, did not detect speech at all. It computed the RMS level of every 10 ms window and slid the binary subtitle activity over it, taking the offset with the largest dot product. Explosions, score and crowds are loud too, and a loudness curve cannot tell them from a conversation. A second attempt summed FFT magnitudes over the telephone band, 300 to 3 400 Hz, per window, and was replaced within days by libfvad in its least aggressive mode, giving a binary decision per frame. That version downsampled to 8 kHz by keeping every $\lfloor f_s/8000 \rfloor$-th sample. For 48 kHz audio this only aliased; for 44.1 kHz it kept every fifth sample, which is 8 820 Hz, and the whole timeline came out 9% too long. It also decoded only the front left channel. Proper resampling through libswresample fixed both on 6 August, and the channel selection of Section 5.2 came with it, together with Silero at 8 kHz with 256 sample windows. Version 2.0.0 moved Silero to 16 kHz and added the lanes. The pieces were one per CPU core until 2.0.0, which put the detector's warm up at different places on a 4 core and a 16 core machine and gave slightly different answers on each; fixed pieces made Proposition 4.1 true. A development build before 2.0.0 listened to 30 windows spread over the film instead of the whole soundtrack. Once decoding ran in parallel and Silero ran batched, the whole film cost only seconds, and sampling was removed; caches from that build are still recognised by their coverage and completed.

6 Searching for one offset

6.1 The exact sweep

Lemma 3.5 makes S in (6) a sum of KN trapezoids, and a sum of piecewise linear functions is piecewise linear.

Theorem 6.1. *S is continuous and piecewise linear, with breakpoints contained in the set $\mathcal{B}$ of the $4KN$ trapezoid breakpoints. If $S \not\equiv 0$, its maximum over $\mathbb{R}$ is attained at a point of $\mathcal{B}$, and in particular at an integer.*

Proof. Continuity and piecewise linearity are inherited from each ϕ_{kn}. Between two consecutive points of $\mathcal{B}$ the function is linear, so its maximum over the closed segment is at an end. Outside $[\min \mathcal{B}, \max \mathcal{B}]$ every term is zero. Every segment's maximum is therefore at a breakpoint, and so is the global maximum. The breakpoints are integers by Lemma 3.5. □

Corollary 6.2. *Searching integer offsets loses nothing:* $\max_{\sigma \in \mathbb{Z}} S(\sigma) = \max_{\sigma \in \mathbb{R}} S(\sigma)$.

The sweep of Algorithm 4 evaluates S at every breakpoint by accumulating slope changes. Two details matter for long films. Pairs whose trapezoid lies entirely outside the search range $[-D, D]$ are skipped before sorting, which is exact, since they contribute nothing inside the range. And the best value in every 1 s bucket is kept, for the noise estimate of Section 7.7.

Algorithm 4. Exact single offset search

Input: spans c, weighted spans (r, h), range D
Output: $\hat{\sigma} = \arg\max_{|\sigma| \le D} S(\sigma)$ over breakpoints, bucket maxima

```
E ← ∅
for every pair (k, n) with r_1^k − c_2^n ≤ D and r_2^k − c_1^n ≥ −D do
    s ← h_k / max(len r^k, len c^n);   add (β_1, +s), (β_2, −s), (β_3, −s), (β_4, +s) to E   ▷ Lemma 3.5
end for
sort E by position
v ← 0; slope ← 0; last ← 0
```

```
for (β, Δ) ∈ E in order do
    v ← v + slope · (β − last)                                   ▷ v = S(β)
    if |β| ≤ D then update the best (v, β) and the maximum of bucket ⌊(β + D)/1000⌋
    end if
    slope ← slope +Δ;   last ← β
end for
```

The cost is $O(E \log E)$ with $E \le 4KN$. On a feature film with 1 500 cues and 2 000 speech spans, $4KN = 12$ million, and the range $D = 15$ minutes keeps roughly a quarter of the pairs. A few million events have to be sorted per call, which is fine once and slow for the dozens of calls that drift and slice analysis need. LAPSE therefore runs the common path on FFTs, and keeps the sweep for three jobs: searches wider than the FFT covers (the ±90 minute searches of Section 9.7), references with fewer than four spans, and the per cue score curves of split mode.

6.2 Why the maximum of S is not enough

The sweep finds the maximum of S exactly. The trouble is that S answers a slightly different question from the one we care about. Write the reference as a speech indicator $x(t)$ and the subtitle as $y(t)$, and split each into a local density and a remainder,

$$x = \bar{x} + \tilde{x}, \qquad \bar{x} = b_W * x, \tag{14}$$

with b_W a moving average over a window W of, say, half a minute. Up to normalisation, S behaves like the correlation of x and y, which splits into four terms:

$$x \star y = \underbrace{\bar{x} \star \bar{y}}_{\text{hill}} + \bar{x} \star \tilde{y} + \tilde{x} \star \bar{y} + \underbrace{\tilde{x} \star \tilde{y}}_{\text{spike}}. \tag{15}$$

The first term changes slowly with the lag, on the scale of minutes over which the amount of talking changes. It is large wherever dense parts of the subtitle meet dense parts of the film, whether or not a single line matches. The cross terms are small, because $\tilde{x}$ and $\tilde{y}$ have near zero mean over any window of length W. The last term holds the information about individual lines: it is sharp, a few seconds wide, and peaks only at the correct lag.

On most films the spike is tall enough to win. On a film with a quiet half hour next to a busy one, the hill can be taller than the spike, and the spike sits on its flank. The notes in the source name *The Godfather*, *Saving Private Ryan* and *The Tree of Life* as films where this happened. The FFT search removes the hill before it looks for the peak.

6.3 Two channels on a grid

Time is cut into bins of $G = 40$ ms. For the reference, $n_r = \lfloor r_2^K / G \rfloor + 2$ bins; for the subtitle, as many as its last cue needs.

Definition 6.3 (Channels)**.** For a span a let $B(a) = \{\lfloor a_1/G \rfloor, \dots, \lfloor a_2/G \rfloor\}$ be the bins it touches and $m(a) = |B(a)|$. The *block channels* are

$$u_i^{\mathrm{b}} = \sum_k h_k \, \mathbf{1}[i \in B(r^k)], \qquad v_i^{\mathrm{b}} = \sum_n \frac{\mathbf{1}[i \in B(c^n)]}{m(c^n)}. \tag{16}$$

The *onset channels* put an impulse at the first bin of each span and smooth it with a Gaussian kernel κ of standard deviation 120 ms (3 bins), truncated at three standard deviations and normalised to sum 1:

$$u^{\mathrm{o}} = \kappa * \sum_k h_k \, \delta_{\lfloor r_1^k / G \rfloor}, \qquad v^{\mathrm{o}} = \kappa * \sum_n \delta_{\lfloor c_1^n / G \rfloor}. \tag{17}$$

Each cue carries a total mass of one in the block channel, however long it is, so a long cue does not outvote a short one. The correlation at lag j is $(u \star v)_j = \sum_i v_i \, u_{i+j}$, and a positive j means the cues should move later by jG.

Proposition 6.4 (What the block channel measures)**.** *Ignoring the rounding of span ends to bins,*

$$(u^{\mathrm{b}} \star v^{\mathrm{b}})_j = \sum_n \sum_k h_k \, \frac{\mathrm{ov}(r^k,\, c^n + jG)}{\mathrm{len}\, c^n}, \tag{18}$$

the overlap score normalised by the cue's own length instead of by the longer of the two spans.

Proof. $\sum_i v^{\mathrm{b}}_i u^{\mathrm{b}}_{i+j} = \sum_n \frac{1}{m(c^n)} \sum_k h_k \, |\, (B(c^n) + j) \cap B(r^k) \,|$. The count of shared bins is the overlap in bins, $\mathrm{ov}(r^k, c^n + jG)/G$ up to rounding at each end, and $m(c^n) = \mathrm{len}\, c^n / G$ up to the same rounding. □

The difference from Lemma 3.4 is deliberate. Normalising by the cue length makes the block correlation reward a cue for being covered by speech at all, and leaves the question of how well the lengths match to the exact rescoring in Section 7.1.

Proposition 6.5 (What the onset channel measures)**.** *Treating the kernel as continuous,* $(u^{\mathrm{o}} \star v^{\mathrm{o}})_j = \sum_n \sum_k h_k \, \gamma(r^k_1 - c^n_1 - jG)$*, where* γ *is a Gaussian of standard deviation* $120\sqrt{2} \approx 170$ ms.

Proof. Correlation is bilinear, so the double sum of impulses gives $\sum_{n,k} h_k \, (\kappa \star \kappa)$ evaluated at the distance between the two impulses at lag j. The correlation of a centred Gaussian with itself is a Gaussian whose variance is the sum of the two variances. □

So the onset correlation is a soft count of cue starts that land within a couple of hundred milliseconds of a speech start. Subtitlers put a line up when the speaking starts, so the starts carry more timing information than the coverage does. The block channel is still needed, because the onsets alone are sparse and many films have long stretches where speech starts are ambiguous.

6.4 Removing the trend

Both channels of both sides have a centred running mean subtracted: over $W_{\mathrm{b}} = 30$ s (750 bins) for blocks and $W_{\mathrm{o}} = 8$ s (200 bins) for onsets. Write $g = \delta - b_W$ for this filter, with b_W the centred box average. Away from the first and last $W/2$ of the signal, g is linear and shift invariant, and the following holds.

Proposition 6.6 (Filtering both signals filters the score)**.** *For finite sequences* x, y *and any filter* g *with finite support,*

$$(g * x) \star (g * y) = (g \star g) * (x \star y). \tag{19}$$

In the frequency domain the correlation is multiplied by $|\hat{g}(f)|^2$*.*

Proof. Let $\hat{x}(f) = \sum_i x_i e^{-2\pi \mathrm{i} f i}$. The correlation $(x \star y)_j = \sum_i x_i y_{i+j}$ has transform $\overline{\hat{x}(f)}\, \hat{y}(f)$. Filtering multiplies each transform by $\hat{g}$, so the correlation of the filtered sequences has transform $\overline{\hat{g}\hat{x}}\, \hat{g}\hat{y} = |\hat{g}|^2\, \overline{\hat{x}}\hat{y}$, and $|\hat{g}|^2$ is the transform of $g \star g$. □

For the centred box of odd width W, $\hat{b}_W(f) = \dfrac{\sin(\pi f W)}{W \sin(\pi f)}$, a real Dirichlet kernel with $\hat{b}_W(0) = 1$ and first zero at $f = 1/W$. Hence

$$|\hat{g}(f)|^2 = \left(1 - \frac{\sin(\pi f W)}{W \sin(\pi f)}\right)^2, \tag{20}$$

which is 0 at $f = 0$, grows like $\left(\pi^2 W^2 f^2 / 6\right)^2$ for small f, and is close to 1 for $f \gtrsim 1/W$. Filtering both channels therefore high pass filters the score curve over lags, with a cutoff of about one cycle per W. The hill in (15) lives at lag frequencies below $1/W$ and is suppressed, while the spike, a few seconds wide, has most of its energy above $1/W$ and passes. The same filter makes the curve's mean zero, which is what gives the median based noise estimate in Section 6.8 something to measure against.

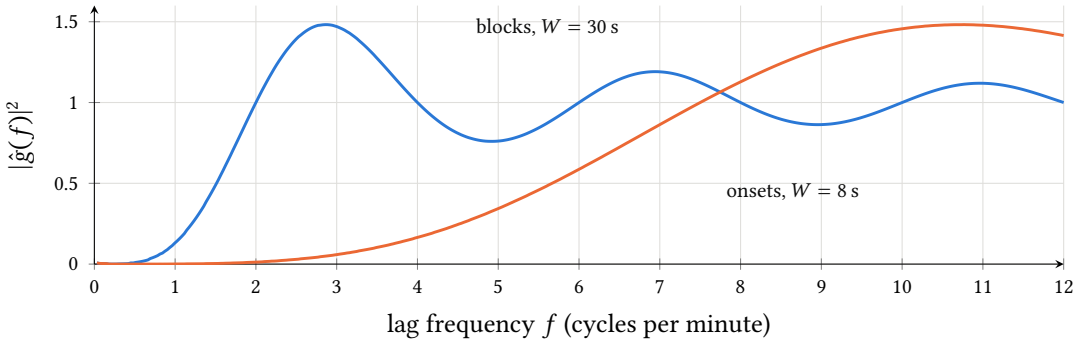

Figure 2. Response of the trend removal on the score curve, (20), with lags in 40 ms bins (1 500 per minute). Anything in the score that changes more slowly than once per 30 s (blocks) or per 8 s (onsets) is removed. Values above 1 come from the side lobes of the box average.

6.5 Normalisation and the combined curve

Each channel is divided by its Euclidean norm over the bins it occupies, so the per channel correlation is

$$\rho^{\mathrm{b}}_j = \frac{(u^{\mathrm{b}} \star v^{\mathrm{b}})_j}{\|u^{\mathrm{b}}\|\,\|v^{\mathrm{b}}\|}, \qquad \rho^{\mathrm{o}}_j = \frac{(u^{\mathrm{o}} \star v^{\mathrm{o}})_j}{\|u^{\mathrm{o}}\|\,\|v^{\mathrm{o}}\|}, \tag{21}$$

with u, v now the filtered channels. By the Cauchy and Schwarz inequality, and since a circular shift preserves the norm, $|\rho_j| \le 1$ for both. The combined curve is $\rho^{\mathrm{b}} + 1.15\,\rho^{\mathrm{o}}$, bounded by 2.15 in absolute value. The weight 1.15 gives the onsets a slight edge, for the reason given after Proposition 6.5.

6.6 Four passes with different blur

No single level of detail is right for every film. A sharp correlation finds the peak on dialogue heavy films and misses it on sparse ones, where the few matching lines are not enough to rise above the noise unless neighbouring lags pool their evidence. The engine therefore reads four curves from the same four forward transforms, each with a Gaussian blur applied in the frequency domain:

Pass	Channels	Blur (standard deviation)
1	blocks and onsets	none
2	blocks and onsets	550 ms
3	onsets only	450 ms
4	blocks only	250 ms

Proposition 6.7. *Multiplying the cross spectrum by $e^{-2\pi^2 s^2 f^2}$, with f in cycles per bin, convolves the correlation curve with a Gaussian of standard deviation s bins, up to the periodisation of the DFT.*

Proof. $\int_{\mathbb{R}} \frac{1}{s\sqrt{2\pi}} e^{-t^2/2s^2} e^{-2\pi i f t}\,dt = e^{-2\pi^2 s^2 f^2}$, and multiplication of transforms is convolution of the sequences. For s far below the transform length the periodised Gaussian is indistinguishable from the Gaussian. □

6.7 Transform length and wrap around

A DFT correlates circularly. The transform length is chosen so that no lag inside the search range wraps.

Proposition 6.8. *Let x be supported on $[0, L_x)$, y on $[0, L_y)$, and $N_{\mathrm{fft}} \ge \max(L_x, L_y) + J$. For $|j| \le J$ the circular correlation at lag j equals the linear one.*

Proof. The circular correlation at lag j sums $x_i y_{(i+j) \bmod N}$ over $i < L_x$. It differs from the linear one only if some $i + j \pm N$ falls in $[0, L_y)$. Since $i + j + N \ge N - J \ge L_y$ and $i + j - N < L_x + J - N \le 0$, neither can. □

With $J = 15\,\text{min}/G = 22\,500$ bins, the engine takes

$$N_{\text{fft}} = 2^{\lceil \log_2(1.25\, n_r + J + 64) \rceil} \tag{22}$$

and truncates the subtitle channels to $N_{\text{fft}} - J - 8$ bins, which satisfies the proposition on both sides. For a two hour film, $n_r = 180\,000$ and $N_{\text{fft}} = 262\,144$. The factor 1.25 leaves room for a subtitle that runs longer than the speech in the film, which is exactly the case of a file timed at 23.976 against a 25 fps video.

6.8 Peaks and their height

From each pass the engine takes the ten highest peaks, greedily, excluding ±2.5 s around each one taken, and scores each by how far it stands out of the curve's noise:

$$z = \frac{x_{\text{peak}} - \text{med}(x)}{1.4826\ \text{MAD}(x)}, \tag{23}$$

with the median and the median absolute deviation computed over every third point of the curve. A peak itself does not move the median, and the MAD ignores the handful of large values around the peak, so z measures the peak against the bulk of the curve and not against its own shoulders [7]. The candidates from the four passes are pooled, sorted by z, and thinned so that no two lie within 2 s of each other. Up to 40 remain. Algorithm 5 collects the steps.

Algorithm 5. FFT candidate search

Input: reference transforms $\hat{u}^{\text{b}}, \hat{u}^{\text{o}}$ and norms (computed once per film); cues c
Output: candidate offsets with scores z

build $v^{\text{b}}, v^{\text{o}}$ from c; subtract running means; compute norms and $\hat{v}^{\text{b}}, \hat{v}^{\text{o}}$
for each pass $(\alpha_{\text{b}}, \alpha_{\text{o}}, s)$ in the table of Section 6.6 **do**
  $X_f \leftarrow \left(\alpha_{\text{b}}\, \hat{u}^{\text{b}}_f \overline{\hat{v}^{\text{b}}_f} / \|u^{\text{b}}\| \|v^{\text{b}}\| + 1.15\, \alpha_{\text{o}}\, \hat{u}^{\text{o}}_f \overline{\hat{v}^{\text{o}}_f} / \|u^{\text{o}}\| \|v^{\text{o}}\|\right) e^{-2\pi^2 s^2 (f/N_{\text{fft}})^2}$
  $x \leftarrow$ inverse transform of X, restricted to lags $|j| \le J$
  take the 10 highest peaks, ±2.5 s apart; score each by (23)
end for
pool, sort by z, drop any within 2 s of a better one, keep at most 40

Per query this costs two forward and four inverse real transforms of length N_{fft}, plus $O(N_{\text{fft}})$ work per peak. For a two hour film it replaces the few million sorted events of the sweep with six transforms of 262 144 points.

> **Design history.** The first search, on 3 July, tried every offset in ±120 s at 10 ms against the loudness curve: 24 001 dot products over the length of the film, around 10^{10} multiply and add operations for two hours, and several minutes per film. On 13 July it became an FFT cross correlation, per 15 minute chunk, which brought a film down to a few seconds. Both profiles were still ones and zeros, and both still carried their mean, so the correlation mostly measured how much talking there was in the overlap. Removing the mean fixed that for a single chunk on 6 August. Removing a running mean, in 2.0.0, fixed it for films whose density changes, which turned out to be the more important version of the same idea. The sweep of Algorithm 4 arrived on 4 August and was the default search from 5 to 10 August. Its first implementation summed the fractional slopes into integer variables; every slope was truncated to zero, the score never left zero, and the default mode returned an offset of 0 for every file until the fix on 6 August.

7 Refinement and the lock statistic

The FFT search is fast and resolves offsets to 40 ms. It is also a different score from S, and its z values are not comparable between passes or films. This section describes how candidates are refined on the exact score and then ranked by a statistic that reads on the same scale for every film.

7.1 Refinement on the exact score

A candidate σ_0 is refined by evaluating S on $\sigma_0 + \{-1200, -1195, \dots, 1200\}$ and then on the 13 integers within ±6 ms of the best point found. Evaluating S at one offset does not need the sweep: the reference is valid, so for each cue a binary search finds the first reference span that can reach it, and only the few spans that overlap it are visited. One evaluation costs $O(N \log K)$, and one refinement 494 of them.

Lemma 7.1 (Slope bound). *Wherever S is differentiable,*

$$|S'(\sigma)| \le \Lambda_S := 2 \sum_{n=1}^{N} \frac{1}{\operatorname{len} c^n}. \tag{24}$$

Proof. By the proof of Lemma 3.5, ϕ_{kn} has a non-zero derivative only when exactly one end of the cue lies strictly inside r^k: the cue's start and not its end, or its end and not its start. Since the reference spans do not overlap, at most one of them contains the start of $c^n + \sigma$ and at most one contains its end. Each cue therefore has at most two pairs with a non-zero derivative, and each such derivative has absolute value $h_k / \max(\operatorname{len} r^k, \operatorname{len} c^n) \le 1/\operatorname{len} c^n$. □

Proposition 7.2 (Grid loss). *Let $\sigma^\star$ maximise S on an interval, and let the grid Σ have step δ and cover the interval. Then* $\max_{\sigma \in \Sigma} S(\sigma) \ge S(\sigma^\star) - \Lambda_S \, \delta/2$.

Proof. Some grid point lies within $\delta/2$ of $\sigma^\star$, and Lemma 7.1 bounds how far S can fall over that distance. □

For a film with 1 500 cues of two seconds on average, $\Lambda_S \approx 1.5$ per millisecond, so the 5 ms scan loses at most about 4 units of score against a typical maximum of several hundred, and the second scan at 1 ms usually recovers the rest. By Corollary 6.2 the second scan finds the exact maximum whenever it lies within 6 ms of the coarse winner, which is the case unless S has two nearly equal maxima more than 5 ms apart. That is not guaranteed, and the engine does not need it to be.

7.2 Bunching

Speech starts carry most of the timing information (Proposition 6.5), and the lock statistic measures them directly. For a cue start c_1^n shifted by σ, let $d_n(\sigma)$ be the distance to the nearest start of a reference span. The *bunching* of an offset is

$$B(\sigma) = \frac{1}{N} \sum_{n=1}^{N} \exp\!\Big(-\frac{d_n(\sigma)^2}{2\tau^2}\Big), \qquad \tau = 450\ \text{ms}, \tag{25}$$

a number in $[0, 1]$ that is 1 when every cue starts exactly on a speech start. The tolerance is wider than the 170 ms of the onset channel, because subtitlers lead or lag the speech by a few frames and speech detectors are late on soft onsets. The second ingredient is the exact overlap per cue, $O(\sigma) = S(\sigma)/N$.

7.3 A baseline from random offsets

Neither B nor O means much on its own. A talky film with a dense reference gives high values at any offset. The lock statistic compares them with the same file at offsets that are certainly wrong. Draw $\xi_1, \dots, \xi_{96}$ uniformly from the integers in $[-900\,000,\ 900\,000]$, with a fixed seed, and set

$$\mu_B = \frac{1}{96} \sum_i B(\xi_i), \quad s_B^2 = \frac{1}{96} \sum_i \big(B(\xi_i) - \mu_B\big)^2, \qquad z_B(\sigma) = \frac{B(\sigma) - \mu_B}{s_B}, \tag{26}$$

and the same for O. The baseline depends only on the cue set, so it is computed once per set and cached.

Proposition 7.3 (Invariance). *z_B does not depend on the weights h_k. z_O is unchanged when every h_k is multiplied by the same $\lambda > 0$, and both are unchanged when the subtitle and the reference are moved by the same amount of time.*

Proof. B uses only span starts. Scaling every h_k by λ scales $O(\sigma)$ by λ at every offset, so it scales μ_O and s_O by λ and leaves $(O - \mu_O)/s_O$ as it was. A common shift of both sequences leaves every overlap and every distance between starts unchanged. □

The first two statements are what make the statistic comparable across films: a detector that is systematically more or less confident, or a film that is simply louder, does not move it. The same cannot be said of S or of the confidence S/N.

Estimation error. The baseline is estimated from a finite sample. The mean of 96 draws has a standard error of $s/\sqrt{96} \approx 0.10\,s$, which moves any z by about ± 0.1. The standard deviation is the larger source: under a normal model its relative standard error is about $1/\sqrt{2 \cdot 95} \approx 7.3\%$, so a true z of 8 is estimated anywhere between roughly 7.4 and 8.6. For the slices, with 48 draws each, the figure is about 10%. The fixed seed makes this error repeatable from run to run, which is useful for users and does nothing for accuracy.

7.4 Slices

A wrong offset can look good on the whole file by fitting one long scene very well. The right offset should look reasonable everywhere. The cue list is cut into eight slices of equal count, and each slice gets its own baseline from 48 random offsets. With $z_{B,j}$ and $z_{O,j}$ the scores of slice j,

$$\zeta_j(\sigma) = \operatorname{clip}\Big(\tfrac{1}{2}\big(z_{B,j}(\sigma) + z_{O,j}(\sigma)\big),\ -1,\ 3\Big). \tag{27}$$

The upper clip keeps one excellent slice from carrying seven poor ones. The lower clip keeps a slice without any dialogue, where every offset is equally bad, from sinking an answer the rest of the file supports. Files with fewer than 64 cues skip the slice term.

7.5 The statistic

Definition 7.4 (Lock statistic)**.** With $z^+ = \max(0, z)$,

$$\Lambda(\sigma) = 0.6\,\big(z_B^+(\sigma) + z_O^+(\sigma)\big) + 1.6 \cdot \frac{1}{8}\sum_{j=1}^{8} \zeta_j(\sigma). \tag{28}$$

It is reported as `sigma`, clipped below at zero.

Proposition 7.5. *$\Lambda \geq -1.6$, and the slice term lies in $[-1.6,\ 4.8]$. For Λ to reach the solid threshold of 8 with the slice term at its maximum, the whole file needs $z_B^+ + z_O^+ \geq 5.3$; with the slice term at zero it needs 13.3.*

Proof. The first term is non-negative. Each $\zeta_j \in [-1, 3]$, so their mean is too, and 1.6 times that interval is $[-1.6, 4.8]$. The two thresholds are $(8 - 4.8)/0.6$ and $8/0.6$. □

The proposition says how the two halves share the work. A file that agrees with itself in every slice needs only a moderate global score; a file where the slices say nothing needs an extreme one. The weights 0.6 and 1.6, the clip limits and τ were set by hand on the benchmark films (Section 14).

Λ is not a z score and not a p value. If z_B and z_O were independent standard normal variables under a wrong offset, a value of 13.3 for their sum would have a probability far below 10^{-30}. They are neither independent nor normal: the random offsets sample one film whose structure repeats, and the same few scenes influence both scores. The threshold was chosen from outcomes, not from a distribution.

7.6 The complete search

Algorithm 6 is the function the rest of the engine calls whenever it needs an offset for a set of cues.

Algorithm 6. Best single offset

Input: cues c, reference (r, h), range D
Output: offset $\hat{\sigma}$, lock $\Lambda(\hat{\sigma})$, confidence, margin

if the FFT is set up and $D \le 15$ min **then**
  $\mathscr{C} \leftarrow$ FftCandidates(c) $\cup\{0\}$ ▷ Algorithm 5; 0 lets an already correct file win
  rank $\mathscr{C}$ by Λ; keep the best four
  for each kept σ_0 **do**
    $\sigma \leftarrow$ Refine(σ_0) on S ▷ Section 7.1
    keep σ if $|\sigma| \le D$ and $\Lambda(\sigma)$ is the best so far
  end for
  return $\hat{\sigma}$, $\Lambda(\hat{\sigma})$, $\min(1, S(\hat{\sigma})/N)$, 1− runner up ratio
else
  return the sweep of Algorithm 4, with Λ replaced by the bucket statistic of Section 7.7
end if

Ranking twice has a reason. Refinement costs 494 evaluations of S per candidate, too many to spend on all 41, so Λ at the unrefined offsets picks the four worth refining; the unrefined offsets are only good to 40 ms, so Λ is computed again after refinement to pick the winner.

7.7 The statistic on the sweep path

When the sweep is used, the lock is measured differently. The search range is cut into 1 s buckets and the maximum of S in each is kept. With M the global maximum and the buckets within ±2 of the winner excluded,

$$\Lambda_{\text{sweep}} = \text{clip}\Big(\frac{M - \text{med(bucket maxima)}}{1.4826\ \text{MAD(bucket maxima)}},\ 0,\ 99\Big). \tag{29}$$

This is the z of (23) applied to the upper envelope of S. It reads on a different scale from Λ, a point that matters for the restart and joined modes of Section 9.7, which use it.

> **Design history.** The first confidence measure, in July, was the best score divided by the mean score over all offsets. It depended on the length of the search range and could not be compared across films. The drift code of version 2.1 used the ratio of the best to the second best score per chunk, with the runner up taken from the lag next to the peak; since a correlation is smooth, that lag is always nearly as high, and every chunk scored about 1. The runner up was then taken from at least a second away. The margin between the two best peaks, 1 − runner up/best, is still computed and reported, and it still appears in the signature of the verdict function, but the verdict ignores it; slice agreement covers the case it was meant for, a peak that fits one part of the film and not the rest. Λ replaced the ratio based measures in 2.0.0.

8 Drift

8.1 The ratios that occur

Under the drift model (2), a subtitle timed for one frame rate and played at another is stretched by the ratio of the two rates. Only a few rates are in use, so only a few ratios occur. Table 2 lists the set P that LAPSE searches.

Pair of rates	ρ	$\rho - 1$	Drift over 2 hours
same rate	1.000000	0	0
24 and 23.976, or 30 and 29.97	1.001001	+0.10%	7.2 s
25 and 24	1.041667	+4.17%	5 min 0 s
25 and 23.976	1.042709	+4.27%	5 min 7 s
30 and 25	1.200000	+20.0%	24 min

Table 2. Framerate ratios in P. Each ratio above 1 also appears as its inverse, which gives eleven entries in the code; 24/23.976 and 30/29.97 are the same number, 1000/999.

Two pairs in the table lie close together. The ratio 1000/999 is 0.1% from 1, and 25/24 and 25/23.976 differ by 0.1% as well. Any method that estimates ρ has to separate values that close.

8.2 Why measuring the lag chunk by chunk fails

The natural approach, and the one LAPSE started with, cuts the film into chunks, finds the lag in each, and fits a line through the lags. It fails for the large ratios.

Proposition 8.1. *Over a chunk of length T_c, a drift of $\varepsilon = \rho - 1$ moves the correct lag by εT_c. If the correlation peak of a single well aligned stretch has width w and $\varepsilon T_c \gg w$, the peak of the chunk's correlation is lowered by a factor of roughly $w/(\varepsilon T_c)$ compared with a chunk without drift.*

Sketch. Each part of the chunk contributes a peak of width about w at its own lag, and the lags spread evenly over an interval of length εT_c. The chunk's correlation is the sum of these shifted peaks, whose total mass is fixed; spread over εT_c instead of w, the height falls by the ratio of the two widths. □

For $\rho = 25/24$ and chunks of 15 minutes, $\varepsilon T_c = 37.5$ s. With a peak width of about two seconds, the chunk's peak is some twenty times lower than it would be without drift, and it sinks into the noise. Short chunks help with the smear but have too few cues to lock. A short list of known ratios avoids the dilemma.

8.3 Searching the ratios

Algorithm 7 scales the cues by each ratio in turn, runs the full offset search, and keeps the ratio with the highest lock. Choosing by the raw score or by the confidence S/N favours the ratio that smears the cues widest, since a stretched file covers more of the reference; Λ compares each ratio with its own random baseline and does not have that bias.

Algorithm 7. Ratio search

Input: cues c, reference (r, h)
Output: ratio $\hat{\rho}$, offset $\hat{\sigma}$

```
for ρ ∈ P do
    (σ_ρ, Λ_ρ) ← BestOffset(ρc)        ▷ Algorithm 6
end for
return ρ̂ = arg max_ρ Λ_ρ and its offset
```

The explicit `o1s` mode uses this search, and falls back to the chunked regression of Section 8.5 only when no ratio reaches a confidence of 0.5, which leaves a path for a stretch that is not on the list. Auto mode uses a cheaper test first.

8.4 The slope from eight slices

Auto mode already has the offsets of eight slices of the file, each found on its own (Section 10.1). Write them as (t_j, σ_j), with t_j the mean midpoint of the cues in slice j. Under the drift model the points lie on the line $\sigma = (\rho - 1)t + \sigma_0$. The slope is estimated with the Theil and Sen estimator,

$$\hat{\varepsilon} = \operatorname{med}\left\{ \frac{\sigma_j - \sigma_i}{t_j - t_i} \;:\; i < j \right\}, \qquad \hat{\sigma}_0 = \operatorname{med}_j\left(\sigma_j - \hat{\varepsilon} t_j\right). \tag{30}$$

Its breakdown point is $1 - 1/\sqrt{2} \approx 29\%$ [11]: with eight slices, about two can lock onto nothing without moving the line far. A slice that lands in a stretch without dialogue does exactly that, so this matters more than efficiency.

Proposition 8.2 (Resolution)**.** *If every slice offset is within η of the line and the slice midpoints span a time D_t, the pairwise slope from the two outermost slices is within $2\eta/D_t$ of ε.*

Proof. $\left| \frac{\sigma_j - \sigma_i}{t_j - t_i} - \varepsilon \right| = \frac{|(\sigma_j - \varepsilon t_j - \sigma_0) - (\sigma_i - \varepsilon t_i - \sigma_0)|}{t_j - t_i} \le \frac{2\eta}{D_t}$. □

For a two hour film the outer slice midpoints are about $\frac{7}{8} \cdot 7200 = 6300\,\mathrm{s}$ apart. With $\eta = 0.4\,\mathrm{s}$, the agreement tolerance of auto mode, the bound is 1.3×10^{-4}, an order of magnitude below the 10^{-3} that separates the closest ratios in Table 2. The median of all pairwise slopes is less precise than the outermost pair in the best case and far more reliable in the worst. Auto mode calls the file drifting when at least four slices lie within 400 ms of the fitted line and $|\hat{\varepsilon}| > 10^{-5}$, then snaps $1+\hat{\varepsilon}$ to the nearest ratio in P if one lies within 0.004. It then scales the cues by the ratio and searches again, and keeps the stretch only if the lock improves by at least one unit: $\Lambda_\rho \geq \Lambda_1 + 1$. A stretch that does not pay for itself is dropped in favour of a plain shift.

8.5 The chunked regression

The fallback in `ols` mode is the original drift method, with its bugs fixed. The reference and subtitle activity profiles, one value per 10 ms, are cut into chunks of 15 minutes starting at 7.5 minutes. Each chunk has its mean removed and is correlated by FFT over lags of ±15 minutes. The peak at lag j is refined by fitting a parabola through its neighbours,

$$\hat{j} = j + \operatorname{clip}\Big(\frac{1}{2} \cdot \frac{x_{j-1} - x_{j+1}}{x_{j-1} - 2x_j + x_{j+1}},\ -\tfrac{1}{2},\ \tfrac{1}{2}\Big), \tag{31}$$

and its sharpness q is the ratio of the peak to the highest value at least a second away. Chunks with $q < 1.05$ are dropped, and a line $\sigma = \varepsilon t + \sigma_0$ is fitted to the rest by weighted least squares with weights q:

$$\hat{\varepsilon} = \frac{\sum q \sum qt\sigma - \sum qt \sum q\sigma}{\sum q \sum qt^2 - \left(\sum qt\right)^2}, \qquad \hat{\sigma}_0 = \frac{\sum q\sigma - \hat{\varepsilon} \sum qt}{\sum q}. \tag{32}$$

When fewer than two chunks survive, or the denominator vanishes, the result falls back to a flat offset instead of dividing by zero.

> **Design history.** Version 2.1, on 8 July, introduced the chunked regression as the only drift method. Three bugs kept it from working. The time of each chunk was computed as $(k-\frac{1}{2})\cdot 900$ s, forgetting that the chunks start 7.5 minutes in. The sharpness took its runner up from the lag next to the peak and came out at about 1 for every chunk. And a wide search of ±120 s meant for the busiest chunk compared a chunk counter with a sample index, so it never ran and every chunk was searched only ±10 s around zero. The fixes on 6 August made the regression run as intended, and on a film converted between frame rates it was still about 67 s off. The ratio search replaced it as the default in the same change; Proposition 8.1 explains why the regression could not have been rescued for the large ratios.

9 Split alignment

A recut film, a television capture with ad breaks, or a director's cut against a theatrical subtitle needs an offset that changes along the file. This section solves (7) on a grid, then cleans up the solution, and finally treats two structures that look like splits but have simpler answers: files whose timestamps start over, and videos that hold two parts.

9.1 The grid formulation

Offsets are restricted to a grid around an anchor σ_0, normally the global offset from Algorithm 6:

$$\Sigma = \{\sigma_0 - W + m\delta \ :\ m = 0, \dots, M-1\}, \qquad M = 2W/\delta + 1. \tag{33}$$

Two grids are used. The recut mode searches $W = 600\,\mathrm{s}$ at $\delta = 100\,\mathrm{ms}$, so $M = 12\,001$. The refinement inside the shifted and drifting modes searches $W = 45\,\mathrm{s}$ at $\delta = 25\,\mathrm{ms}$, so $M = 3\,601$. For each cue n, the per cue score $f_n(m) = \sum_k \phi_{kn}(\sigma_0 - W + m\delta)$ is evaluated exactly at every grid point by a sweep over the reference spans the cue can reach, found by binary search.

9.2 Keeping cues in order

A split must not make lines swap places or overlap more than they did.

Lemma 9.1 (Order constraint). *Shifting cue $n-1$ by σ_{n-1} and cue n by σ_n keeps them in order if and only if $\sigma_{n-1} - \sigma_n \le \gamma_n$, where $\gamma_n = c_1^n - c_2^{n-1}$ is the gap between them.*

Proof. The shifted cues are in order when $c_2^{n-1} + \sigma_{n-1} \le c_1^n + \sigma_n$, which rearranges to the inequality. □

On the grid the constraint becomes $m_{n-1} \le m_n + g_n$ with g_n the gap in grid steps. The code computes g_n by integer division, which truncates towards zero. For non-negative gaps that is the floor and the constraint is exact. For the rare negative gap, which occurs in split mode because cues there are not merged, truncation lets the pair overlap by less than one grid step more than it already did.

9.3 The recursion

Definition 9.2. Let $V_n(m)$ be the best value of $\sum_{i \le n} f_i(m_i) - p \cdot \#\{i \le n : m_i \ne m_{i-1}\}$ over all $m_1, \dots, m_n$ with $m_n = m$ that satisfy the order constraint for every $i \le n$.

Theorem 9.3. *$V_1 = f_1$, and for $n \ge 2$*

$$V_n(m) = f_n(m) + \max\Big(V_{n-1}(m) \cdot [\, g_n \ge 0 \,],\ \max_{m' \le m + g_n} V_{n-1}(m') - p \Big), \tag{34}$$

where the bracket means the first option is only available when $g_n \ge 0$. The optimum of the grid problem is $\max_m V_N(m)$, and the maximising sequence is recovered by following the choices backwards.

Proof. Take an optimal sequence ending at $m_n = m$ and look at m_{n-1}. Either $m_{n-1} = m$, which the order constraint allows exactly when $g_n \ge 0$ and which adds no split; or $m_{n-1} = m' \ne m$ with $m' \le m + g_n$, which adds one split. In both cases the prefix $m_1, \dots, m_{n-1}$ must itself be optimal among sequences ending at m_{n-1}, since otherwise replacing it would improve the whole sequence without affecting the constraint at n, which only involves m_{n-1} and m_n. The prefix therefore contributes $V_{n-1}(m_{n-1})$. The second maximum ranges over all $m' \le m + g_n$, including $m' = m$; that term equals $V_{n-1}(m) - p$, which the first option dominates whenever it is available, so including it changes nothing. □

Proposition 9.4 (Cost). *The recursion runs in $O(NM)$ time, plus the cost of the per cue score curves, and needs $O(NM)$ memory for the back pointers.*

Proof. For fixed n, the sets $\{m' : m' \le m + g_n\}$ grow with m, so the inner maximum is a running prefix maximum over V_{n-1}: advancing m by one adds at most one new element. Each row therefore costs $O(M)$. One back pointer per cue and grid point is needed to recover the sequence. □

For a film with 1 500 cues, the recut grid needs 1 500 × 12 001 back pointers of four bytes, about 72 MB. At a 1 ms step the same window would need 7.2 GB, which is what the first implementation tried to allocate. Algorithm 8 gives the full procedure.

Algorithm 8. Split alignment

Input: cues $c^1..c^N$ in file order, reference, penalty p, anchor σ_0, window W, step δ
Output: an offset per cue

$V \leftarrow f_1$
for $n = 2, \dots, N$ **do**
  $g \leftarrow \mathrm{trunc}\big((c_1^n - c_2^{n-1})/\delta\big)$; best $\leftarrow -\infty$; $a \leftarrow -1$
  for $m = 0, \dots, M-1$ **do**
    while $a < \min(m + g, M - 1)$ **do** $a \leftarrow a + 1$; update best, best$_{\text{arg}}$ with $V[a]$
    end while
    if $m + g < 0$ **then** $V'[m] \leftarrow -\infty$
    else if $g \ge 0$ and $V[m] \ge$ best $- p$ **then** $V'[m] \leftarrow f_n(m) + V[m]$; back$[n][m] \leftarrow m$ ▷ stay
    else $V'[m] \leftarrow f_n(m) +$ best $- p$; back$[n][m] \leftarrow$ best$_{\text{arg}}$ ▷ split
    end if

```
    end for
    V ← V′
end for
follow back pointers from arg max V; convert grid indices to offsets
RefineRuns; SettleBoundaries                                  ▷ Section 9.4
```

9.4 Refining runs and settling boundaries

The grid solution is good to one grid step and has two weaknesses, both addressed after backtracking.

Runs. Each maximal run of cues with the same offset is shifted by that offset and passed once more to the single offset search. If the correction is at most two grid steps, it is applied to the whole run. This recovers the precision the grid gave up, without letting one run wander to a different peak.

Boundaries. Inside a stretch without dialogue every boundary position scores the same, because the cues there touch no speech at either offset. The recursion then places the boundary wherever its tie breaking puts it, which can be in the middle of a conversation on the other side of the silence. For every boundary between a run at offset b and a run at offset a, Algorithm 9 considers moving it up to $R = 60$ cues either way and evaluates

$$J(j) = \sum_{k=i-R}^{j-1} \psi_k(b) + \sum_{k=j}^{i+R} \psi_k(a), \tag{35}$$

where $\psi_k(\sigma) = \sum_\ell \phi_{\ell k}(\sigma)$ is the exact score of cue k.

Proposition 9.5. *The procedure finds the maximiser of J over the window with $O(R)$ evaluations of ψ, and among maximisers it picks the one with the widest gap between cue $j-1$ and cue j.*

Proof. $J(j+1) - J(j) = \psi_j(b) - \psi_j(a)$, so walking the boundary one cue to the right changes the running total by a single difference, and the same holds to the left with the sign reversed. The walk visits every j in the window once and keeps the best value, and it replaces the incumbent on an equal value (within 10^{-9}) when the gap is wider. □

The tie break encodes a fact about films: a recut happens at a scene change, and scene changes tend to sit in the longest pauses. The boundary step does not re-check the order constraint. A moved boundary can put two cues out of order when the offsets on either side differ by more than the gap between those cues; the tie break towards the widest gap makes that less likely without ruling it out.

Algorithm 9. Settling one boundary

Input: offsets per cue with a change at cue i (b before, a after), window R

```
J ← Σ_{k=i−R}^{i−1} ψ_k(b) + Σ_{k=i}^{i+R} ψ_k(a);   best ← J;   at ← i;   widest ← c_1^i − c_2^{i−1}
for j = i + 1, … , i + R do                                              ▷ walk right
    J ← J + ψ_{j−1}(b) − ψ_{j−1}(a)
    if J > best, or J = best and c_1^j − c_2^{j−1} > widest then best, at, widest ← J, j, that gap
    end if
end for
walk left from i − 1 to i − R in the same way, with J ← J + ψ_j(a) − ψ_j(b)
give cues before at the offset b and cues from at onwards the offset a
```

9.5 The penalty

Unless a penalty is given on the command line, LAPSE uses

$$p = \max\bigl(3,\ \ln N\bigr). \tag{36}$$

The reasoning is informal. A spurious split lets a stretch of the file choose its own offset, and its gain is the best improvement that noise offers among M alternatives for that stretch. The total score grows with the number of cues, and the number of stretches a file can be cut into grows with it too, so a fixed penalty becomes cheaper, relative to what noise can buy, as files get longer. A penalty proportional to $\ln N$ has the form of the Bayesian information criterion [9], which charges $\frac{1}{2}\ln N$ per parameter; each split adds two parameters, a position and an offset. The analogy is loose. S is not a log likelihood and nothing here makes the choice optimal in a formal sense. In numbers, p is 3 below 21 cues, 6.2 at 500 cues, 7.3 at 1 500 and 8.0 at 3 000, and by the reading of Section 3, a split has to gain the equivalent of that many perfectly placed cues.

9.6 When a split is kept

The shifted and drifting modes of Section 10 run the fine grid ($W = 45$ s) after a solid verdict and keep its result only if it passes a check:

Definition 9.6 (Worth splitting)**.** A split solution with anchor σ_0 is kept if it has at least two runs, every run has at least 12 cues, every offset lies within 60 s of σ_0, and the largest and smallest offsets differ by at least 120 ms.

Each condition removes a failure seen in practice: a single stray cue given its own offset, a run that jumped to a distant peak, and a split that only moves the file by a frame or two. The limits also bound what this path can find. A cut whose two sides differ by more than 45 s is outside the fine grid, and Section 12.11 shows the cost.

9.7 Restarts and joined parts

Two structures look like splits and are cheaper to solve directly.

Restart. Some subtitle files hold two episodes or two discs pasted together, with timestamps that start over. In file order, before any sorting, a cue that starts more than 60 s earlier than its predecessor marks a restart. Each part is aligned on its own with the sweep over ±90 minutes, and the lock of the file is the lowest lock among the parts. No audio analysis is needed to detect the structure.

Joined parts. A video may hold two parts of a film with increasing timestamps in the subtitle, but with the parts further apart than any recut would move them. Sixteen slices are aligned on their own over ±90 minutes. Among the slices whose confidence times margin is at least 0.02, a jump of more than five minutes between consecutive offsets marks a new part. For each pair of neighbouring parts with offsets σ_a and σ_b, the cut is placed at

$$j^\star = \arg\max_j \Big(S_{\le j}(\sigma_a) + S_{>j}(\sigma_b) \Big), \tag{37}$$

where $S_{\le j}$ scores the cues before j and $S_{>j}$ the rest, each computed with an unweighted merge walk over the two sequences in $O(N+K)$. The scan over j is quadratic in the number of cues, which is acceptable for the one or two cuts such a file has. Each part then gets its own offset from the sweep.

Algorithm 10. Joined parts

Input: cues c in time order, reference
1 $(\sigma_1..\sigma_{16}) \leftarrow$ offsets of 16 slices, each over ±90 min
2 anchors $\leftarrow$ offsets of confident slices where the offset jumps by more than 5 min
3 **if** fewer than two anchors **then return** nothing
4 **end if**
5 **for** consecutive anchors (σ_a, σ_b) **do**
6 $\quad j^\star \leftarrow \arg\max_j S_{\le j}(\sigma_a) + S_{>j}(\sigma_b)$ over cues after the previous cut
7 **end for**
8 align each part on its own; lock $\leftarrow$ lowest part lock

> **Design history.** The split recursion came on 4 August. It evaluated every millisecond, which did not fit in memory for a feature film, and it gave every cue a range of offsets of its own, starting where that cue could first meet the reference. Index m therefore meant a different offset for every cue, the back pointers linked unrelated positions, and the grid indices were never converted back into offsets. The fixes of 6 August introduced the shared coarse grid around one anchor, Σ above. Merging overlapping cues in split mode welded the two halves of a cut file together; since then split mode keeps the cues as they are in the file. The fixed penalty of 6 became $\max(3, \ln N)$ in 2.0.0, and the boundary settling and run refinement were added in the same release.

10 Choosing the correction and judging it

The default mode, `auto`, decides per file which of the models of Section 3 applies, runs the matching search, and ends with a verdict that decides what happens to the file. The explicit modes `nosplit`, `ols` and `split` skip the decision and run one search, but end with the same verdict.

10.1 Evidence from slices

The cue list is cut into eight slices of equal count (fewer when the file has fewer than 40 cues, and none below 10), and Algorithm 6 is run on each slice on its own. The result is eight points (t_j, σ_j) with t_j the mean midpoint of the slice. Three summaries are taken from them, all with the tolerance $\eta = 400$ ms:

$$F = \#\{j \,:\, |\sigma_j - \hat{\sigma}| \le \eta\} \qquad \text{slices that back the global offset,} \tag{38}$$

$$L = \#\{j \,:\, |\sigma_j - \hat{\varepsilon} t_j - \hat{\sigma}_0| \le \eta\} \qquad \text{slices on the Theil and Sen line,} \tag{39}$$

$$C = \text{number of groups of at least two slices} \qquad \text{with consecutive offsets within } \eta. \tag{40}$$

A shifted file puts most slices on the global offset. A drifting file puts them on a sloped line. A recut file puts them in two or more groups, one per version of the offset.

10.2 What agreement is worth

Proposition 10.1. *Suppose that at a wrong offset the eight slice offsets were independent and uniform on the integers in* ± 15 *minutes. Then the probability that at least three of them fall within* $\eta = 400$ ms *of that offset is about* 5×10^{-9}.

Proof. One slice falls within the tolerance with probability $q = 801/1\,800\,001 \approx 4.4 \times 10^{-4}$. The number of slices that do is binomial with parameters 8 and q, and $P(\ge 3) \approx \binom{8}{3} q^3 \approx 56 \times 8.7 \times 10^{-11} \approx 4.9 \times 10^{-9}$. □

The assumption is unrealistic, and the number should be read as an upper end. Slices of one film share its structure, the global offset is chosen partly because the slices support it, and a wrong offset that is wrong because of a repeated scene will draw more than one slice with it. What the proposition does show is why three agreeing slices out of eight is a strong requirement even though it leaves five slices free to fail: a slice that lands in a stretch without dialogue, or on the credits, is common, and five of them in one file are not.

10.3 The decision

Algorithm 11 is the decision procedure. The order of the tests matters: a file is called shifted if it can be, drifting only if the slices lie on a line and not on a level, and recut only if neither holds.

Algorithm 11. Mode selection in `auto`

Input: subtitle cues, reference

```
if some cue, in file order, starts > 60 s before its predecessor then
    return RESTART: one offset per part (Section 9.7)
end if
(σ̂, Λ1) ← BESTOFFSET(c);   slices ← eight slice offsets; compute F, L, C
if F ≥ 3 then kind ← SHIFTED
```

else if $L \geq 4$ and $|\hat{\varepsilon}| > 10^{-5}$ **then** kind ← DRIFTING, $\rho \leftarrow \text{snap}(1 + \hat{\varepsilon})$
else if $C \geq 2$ **then** kind ← RECUT
else
  $(\rho, \Lambda_\rho) \leftarrow$ RATIOSEARCH ▷ Algorithm 7
  kind ← DRIFTING if $\rho \neq 1$, $\Lambda_\rho \geq \theta$ and $\Lambda_\rho > \Lambda_1 + 2$, else SHIFTED
end if
if kind is RECUT **then**
  try JOINEDPARTS; if it finds parts, **return** JOINED
  return RECUT: SPLITALIGNMENT($\hat{\sigma}$, $W = 600$ s, $\delta = 100$ ms)
end if
if kind is DRIFTING **then**
  $(\sigma_\rho, \Lambda_\rho) \leftarrow$ BESTOFFSET(ρc)
  if $\Lambda_\rho < \Lambda_1 + 1$ **then** kind ← SHIFTED ▷ the stretch does not pay
  else verdict from Λ_ρ and eight new slices of ρc
  end if
end if
if kind is SHIFTED **then** verdict from Λ_1 and F
end if
if verdict is solid **then**
  try SPLITALIGNMENT($W = 45$ s, $\delta = 25$ ms); keep it if worth splitting (Definition 9.6)
end if
return kind, offsets, verdict

The reported mode names the path: `auto/shifted`, `auto/shifted+split`, `auto/drifting`, `auto/drifting+split`, `auto/recut`, `auto/joined` and `auto/restart`. The split penalty is set from the file size (Section 9.5) in every path.

Two properties of the procedure are worth stating plainly, because the evaluation comes back to both. First, in the shifted and drifting paths, the verdict is computed for the global offset *before* the fine split search runs, and the split that follows is kept or dropped by Definition 9.6 alone. Nothing measures whether the split made the file better. Second, the fine search only looks within 45 s of the global offset, so a cut whose two sides are further apart can only be found through the recut path, which requires the slices to fall into separate groups first.

10.4 The verdict

Definition 10.2 (Verdict)**.** With Λ the lock of the chosen answer, A the number of agreeing slices, and θ the confidence threshold (default 8, set with `--confidence`),

$$\text{verdict} = \begin{cases} \textit{nothing} & \text{if } \Lambda < \min(3.5, \theta), \\ \textit{solid} & \text{if } \Lambda \geq \theta \text{ and } A \geq 3, \\ \textit{unsure} & \text{otherwise.} \end{cases} \tag{41}$$

A is F in the shifted path, the slices of the stretched file that agree with the stretched offset in the drifting path, and the slices that agree with the split solution at their midpoints in the recut path. The restart and joined paths use the lowest part lock from the sweep statistic of Section 7.7 and count as agreeing when that lock is at least 4.

The verdict decides what is written.

Verdict	What happens	Exit code
solid	the subtitle is overwritten, the original is kept as `.bak`	0
unsure	the original is untouched, the answer goes to `name.lapse-unsure.ext`	3
nothing	the original is untouched, the best guess goes to the same sidecar	3

`--force` writes every answer as if it were solid, `--strict` writes nothing instead of a sidecar, and `--no-sidecar` writes nothing either way. An existing `.bak` is never overwritten, so the first backup stays the untouched original however many times the engine runs on a file.

The asymmetry is deliberate. A correct answer left in a sidecar costs a user a rename. A wrong answer written over a correct file costs them the file, and on a library scan they will not know which one it was. The thresholds are set so that the first kind of error is more common than the second.

10.5 Snapping to picture cuts

A subtitler working to picture starts a line on the cut. A global shift that is right to within 60 ms keeps every line 60 ms off the cut it was written against, and a line that appears a moment after the cut reads as late. With `--snap`, after the offset is settled, Algorithm 12 moves each cue start that lies within a window of a picture cut onto the cut, and moves its end by the same amount, so the line stays up as long as it did.

Algorithm 12. Snapping cue starts

Input: corrected cue times, keyframe times $\kappa_1 < \cdots < \kappa_Q$, window ω (default 120 ms)

if $Q < 8$, or the median keyframe gap is below 400 ms, or more than half the gaps lie within 100 ms of the median **then**
  return unchanged ▷ no scene detection in this encode
end if
for each cue $[s, e)$ **do**
  $\kappa \leftarrow$ the keyframe nearest to s; $\Delta \leftarrow \kappa - s$
  if $0 < |\Delta| \le \omega$ and $0 \le s + \Delta < e$ **then** move the cue by Δ
  end if
end for

The cuts are the keyframes the encoder already wrote, read from the packet index without decoding a frame, so the step costs hundredths of a second. Encoders put a keyframe at scene changes and also on a timer. When more than half the gaps between keyframes sit within 100 ms of the median gap, the timer is doing all the work, scene detection was off, and there is nothing to snap to; the engine then leaves the file alone and says so. Snapping is off by default, runs only against a video reference, and never changes how the offset itself is found. It does not re-check the order of neighbouring cues after moving them, so two cues closer together than the window can in principle end up overlapping.

11 A worked example

This section follows one film through the engine, three times, with the numbers the engine computed along the way. The film is *(500) Days of Summer* (2009): 95 minutes, an MP4 file with AC-3 stereo audio, and an English SDH subtitle of 1 434 cues for this release. Three broken copies were made from it, and every error below is measured against a copy of the subtitle that the owner corrected by hand and checked against this exact file:

- **shift**: every cue 33 s late;
- **drift**: every time multiplied by 25/23.976, as if the subtitle were timed for a 23.976 fps release and played at 25 fps;
- **cut**: cues before 00:48:00 moved 21 s early and cues after it 33 s late, a cut with a spread of 54 s.

The runs used the release build of LAPSE 2.2.3 on an Apple M2 Pro with ten cores, in the default `auto` mode with `--no-embedded`. The intermediate values below come from a second build of the same commit with extra logging added; its output files were compared with the release build's and are byte for byte identical. The error of a result is measured per cue, as the absolute difference between its start and the start of the same cue in the ground truth.

11.1 Input and speech

The parser read 1 434 cues. Sixty three of them, such as `[Whistling]`, `[Pencil Scribbling Rapidly]` and a line of music marks, were dropped from scoring by the rules of Section 4.5. After merging, 1 371 spans remained, and the split penalty became $p = \ln 1371 = 7.22$.

The track has two channels, so there was no mix to choose. The film was decoded in ten pieces of ten minutes, in parallel, and Silero found 1 269 speech spans. The first run took 6.1 s of wall clock time (22.3 s of CPU time across the cores). The spans were cached, and every later run on this film took between 0.55 and 0.64 s.

11.2 The shifted copy

FFT search. The reference occupies $n_r = 139\,820$ bins of 40 ms, so $N_{\text{fft}} = 262\,144$. The four passes of Algorithm 5 agreed on the region of the answer and differed in the detail:

Pass	Channels, blur	Best peak (ms)	z
1	both, none	−32 720	36.8
2	both, 550 ms	−32 720	26.5
3	onsets, 450 ms	−32 640	27.8
4	blocks, 250 ms	−32 920	24.2

The strongest rival in any pass was −30 440 ms with $z = 8.3$ in pass 4, two seconds from the answer: the same dialogue lined up one line off. After pooling and thinning, 21 candidates remained, including offset zero.

Ranking and refinement. The candidates were ranked by Λ at their unrefined offsets, and the best four were refined on the exact score:

Candidate (ms)	Λ before	Refined (ms)	S	Λ after
−32 720	16.45	−32 871	514.4	**15.39**
−35 240	7.83	−34 034	464.0	3.98
−30 440	6.76	−31 646	456.8	6.36
−62 160	3.62	−62 627	380.1	3.00

Offset zero came fifth with $\Lambda = 2.47$ and was not refined. The two rivals at −34 and −31.6 s score only 10% less on S than the winner, and their Λ is a quarter of it. That gap is the reason Λ picks the winner: the raw score separates the right answer from a neighbouring line by 10%, the lock statistic by a factor of four.

The lock of the winner. At −32 871 ms, the bunching was $B = 0.460$ against a random baseline of $\mu_B = 0.225$, $s_B = 0.019$, so $z_B = 12.2$; the overlap per cue was $O = 0.375$ against $\mu_O = 0.241$, $s_O = 0.021$, so $z_O = 6.3$. The whole file term is $0.6\,(12.2 + 6.3) = 11.11$. The eight slice terms were 2.08, 3, 1.92, 3, 3, 3, 2.74, 2.66, four of them at the clip of 3, with a mean of 2.68 and a contribution of $1.6 \times 2.68 = 4.28$. Together $\Lambda = 15.39$. The confidence, S/N, was 0.375.

Slices and decision. Aligned on its own, each slice of 171 cues gave:

Slice	Midpoint	Offset (ms)	Λ
1	7:02	−32 967	4.6
2	16:58	−32 944	7.4
3	27:22	+756 055	4.2
4	38:59	−32 807	7.6
5	50:55	−32 853	11.2
6	1:02:44	−32 802	8.3
7	1:13:09	−32 690	6.1
8	1:27:15	−32 746	6.0

Slice 3 locked onto nothing and landed more than twelve minutes away. The other seven lie within 181 ms of the global offset, so $F = 7$, the file is SHIFTED, and with $\Lambda = 15.4 \geq 8$ and $7 \geq 3$ agreeing slices the verdict is *solid*. The fine split search then ran with $W = 45$ s, $\delta = 25$ ms and $M = 3\,601$, and returned a single run: nothing to split.

Result. Offset −32 871 ms, written over the file. Every cue ended 21 ms from the ground truth. The error is the same on all 1 434 cues and reappears in the drift run below, so it is a constant difference between the ground truth and the engine's reading of the audio, not noise in the search.

11.3 The drifting copy

Stretched by 4.3%, the file drifts by four minutes over the film, and no single offset fits it. The global search returned −53 310 ms with $\Lambda = 3.57$. The slices scattered over ±15 minutes, none agreed with the global offset, and the Theil and Sen slope through them was −0.24, which no frame rate produces. F and L both fell short, the slices formed no groups, and the engine fell through to the ratio search of Algorithm 7:

Ratio ρ		Offset (ms)	Λ
1		−53 310	3.57
24/23.976, 30/29.97	1.00100	−235 084	3.74
23.976/24, 29.97/30	0.99900	−32 471	3.35
25/24	1.04167	−501 867	2.83
24/25	0.96000	−2 430	8.27
25/23.976	1.04271	−298 153	2.61
23.976/25	**0.95904**	**129**	**16.12**
30/25	1.20000	−197 035	2.56
25/30	0.83333	−305 892	3.47

The correct ratio stands out with $\Lambda = 16.1$, and so does its neighbour: 24/25 differs from it by 0.1%, stretches the file by the right amount to within 5.7 s over the whole film, and reaches $\Lambda = 8.3$, just over the solid threshold and about half the lock of the correct ratio. The ratio passed both tests of Algorithm 11: $16.1 \geq 8$ and $16.1 > 3.57 + 2$. The engine scaled the cues by 0.95904, searched again, found 129 ms with $\Lambda = 16.1$, which beats the unscaled lock by far more than one unit, and aligned eight new slices of the scaled file. Seven agreed, the verdict was *solid*, and the fine split search found nothing.

Result. Ratio 0.95904, offset +129 ms, reported as `auto/drifting`. Every cue again ended 21 ms from the ground truth.

11.4 The cut copy

Slices. Slices 1, 2 and 4, before the cut, found offsets between +21 033 and +21 193 ms. Slices 5 to 8, after it, found between −32 868 and −32 690 ms. Slice 3 again landed nowhere. The slices therefore form two clean groups, the signature of a recut, before any split search has run.

Global offset. The two halves compete for the global answer. After refinement, +21 059 ms reached $\Lambda = 10.21$ and −32 805 ms reached 9.86: two nearly equal locks for two different parts of the file. No single offset describes this file, and the engine split it.

Result in auto mode. Two parts, verdict *solid*. The first half was placed at +21 074 ms and ended 74 ms from the ground truth; the second half was placed at −33 181 ms and ended 181 ms from it.

Forced split. Run again in `split` mode, the recursion searched ±600 s at 100 ms, $M = 12\,001$, in 0.64 s, and returned two parts at +21 068 and −32 812 ms. The 23 cues from 00:48:03 to 00:49:57, the stretch right after the cut, ended within 146 ms of the ground truth.

11.5 What the example shows

Every stage did what the preceding sections describe, and the evidence was not close. In the shifted copy, the lock statistic separated the answer from its best rival by a factor of four where the overlap score separated them by 10%. In the drifting copy, the correct ratio reached twice the lock of its nearest

neighbour, 0.1% away. In the cut copy, the slices found both halves on their own before any split search ran, and both the automatic and the forced split recovered the two parts. One film does not change the benchmark picture of Section 12.11, where automatic splitting fails most often when the two sides of a cut are far apart.

12 Evaluation

All numbers in this section come from the benchmark published in `docs/benchmarks.md` on 11 August 2026, which measured LAPSE 2.0.0 against alass 2.0.0 and ffsubsync 0.5.1. The per film tables in Appendix B reproduce that file, and every count below is taken from its per film rows.[1]

12.1 Questions

The benchmark was built to answer four questions:

Q1 How often does each tool fix a shifted or drifting subtitle on hard material?

Q2 How often does each tool fix a subtitle with cuts, when splitting is on, and when the tool has to decide by itself?

Q3 How long does a run take, and what does the speech cache change?

Q4 When LAPSE says *solid*, how often is it right, and what does it hold back?

Q4 has no counterpart in the other tools, which do not report a verdict.

12.2 Material

Films. The 39 films were taken from a home library for the properties that make synchronization hard, not at random: long running times, long stretches without dialogue, heavy score under the dialogue, and films that exist in more than one cut. The list is in Table 9. Four films (*2001*, *Challengers*, *Come and See*, *Seven Samurai*) were `.mkv` files that also carried embedded subtitles; the other 35 were `.mp4`.

Subtitles. For each film, the most downloaded subtitle on OpenSubtitles for that release was used, in SubRip format. The subtitle text is therefore what a user would get, including sound descriptions, lyrics and whatever timing errors the upload already had.

12.3 Error injection

Two scripts produced the test files from each subtitle.

Normal test. Every cue was shifted by a constant between 12 and 97 s, or the file was stretched by a framerate ratio (1.001, 0.999 and 1.04167 appear in the first round).

Split test. A copy was cut at one to four timestamps, and each segment was given its own offset between −45 and +45 s. Twenty five films had one cut, eight had two, five had three and one had four. The cut points and offsets of the second round are listed in Table 11; the first round cut each film once.

The split test moves segments of the subtitle and leaves the video alone. A real recut removes or adds footage, so some cues have no speech to land on at all. The synthetic cut is the easier of the two for every tool, and Section 14 returns to the difference.

[1] The summary rows at the foot of two tables in `benchmarks.md` disagree with the rows above them by one film. The per film rows give 28 passes for alass and 18 for ffsubsync's split mode in the split test, and 1 pass (*The Seventh Seal*) for ffsubsync's default mode, where the summary rows say 27, 17 and 0. The prose of that file also swaps two verdicts for *Manchester by the Sea*. This report follows the per film rows throughout.

12.4 Tools and configurations

Run	Configuration
LAPSE auto	default mode, penalty chosen by the engine; run twice, the second run reading the speech cache
LAPSE forced	`split` mode, penalty chosen by the engine, cached
alass	as shipped; splitting is always on, penalty 7
ffsubsync	as shipped; no splitting
ffsubsync split	with `--split-penalty`; the value used is not recorded

In the normal test, LAPSE ran in auto mode only, which covers shift and drift. In the split test, the auto run is the one that matters for a library, where nobody knows which files were cut; the forced run answers the question of what the engine can do once it knows.

12.5 Outcome measures

A run *passes* when every cue lands close to where it belongs, and is *partial* when some segments land and others do not, as long as at least half do; below half it fails. The tolerance was judged per film and not recorded as a number, which is the weakest point of the protocol. The first round also recorded the error of each tool in milliseconds, measured against the original download, and the location and number of cuts each tool produced. The second round recorded LAPSE's verdicts. Timings are single runs on one otherwise idle machine; the hardware is not recorded in `benchmarks.md`.

12.6 Statistical treatment

With 39 films, a difference of a few passes can be noise. Pass rates are reported with 95% Wilson score intervals. Differences between two tools are tested on the paired outcomes with an exact McNemar test: if b films pass only with the first tool and c only with the second, the two sided p value is $\min\left(1,\ 2\sum_{k\le\min(b,c)}\binom{b+c}{k}2^{-(b+c)}\right)$. Partial results count as failures in these tests. No correction for multiple comparisons is applied; the tests are few and are reported to temper the raw counts, not to claim significance.

12.7 Shift and drift

	LAPSE 2.0.0	alass 2.0.0	ffsubsync 0.5.1
Passed, of 39	**36**	31	33
95% interval	80 to 97 %	64 to 89 %	70 to 93 %

Table 3. Normal test (shift or drift).

LAPSE failed three films: *Stalker* and the first two *Godfather* films. The two *Godfather* films failed for all three tools. LAPSE was the only tool to pass *Come and See* (alass crashed after 0.09 s, ffsubsync landed 33 s off) and *The Godfather Part III*. alass passed *Stalker*, the one film where LAPSE failed and another tool did not.

On the paired outcomes, LAPSE passed 6 films that alass failed and alass passed 1 that LAPSE failed ($p = 0.125$). Against ffsubsync the split is 3 to 0 ($p = 0.250$). The intervals in Table 3 and Figure 3 overlap. On this sample, LAPSE is ahead in the normal test, but the lead is not large enough to rule out chance.

12.8 Cuts

With splitting on, LAPSE passed 32 films against 28 for alass (7 to 3 on the paired outcomes, $p = 0.344$) and 18 for ffsubsync's split mode (15 to 1, $p = 0.0005$). The second difference is clear; the first is not.

The more important comparison is inside LAPSE. Left to decide by itself, it passed 19 films, against 32 when it was told to split. On the paired outcomes, forcing the split helped on 14 films and hurt

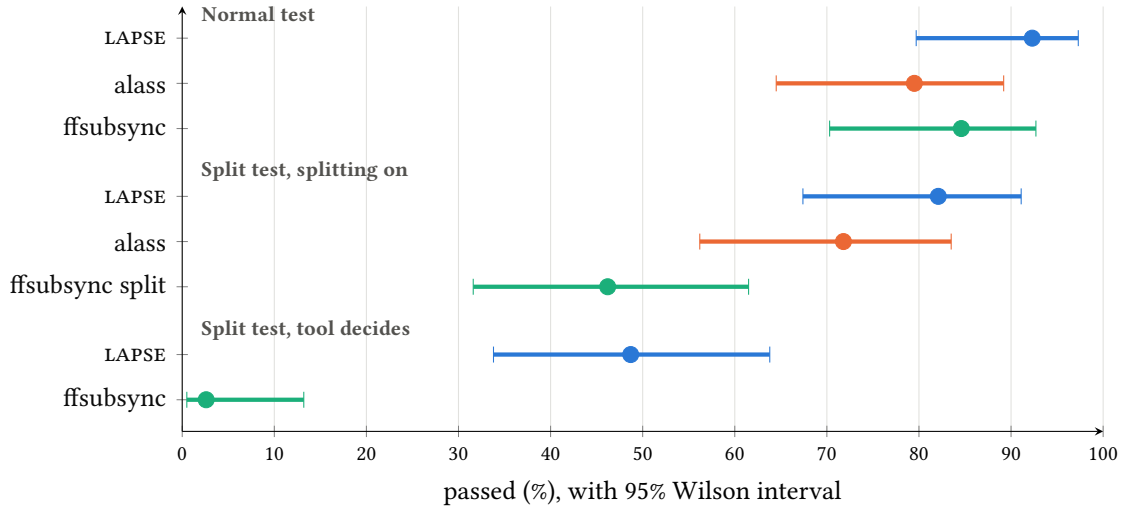

Figure 3. Pass rates on 39 films with 95% intervals. In the bottom group, alass has no entry because it always splits, and ffsubsync does not split unless told to.

	LAPSE auto	LAPSE forced	alass	ffsubsync split
Passed, of 39	19 (+5)	**32** (+3)	28 (+1)	18 (+1)
95% interval	34 to 64 %	67 to 91 %	56 to 83 %	32 to 61 %

Table 4. Split test. Partial results in brackets.

on 1 ($p = 0.0010$). The decision to split, not the split itself, is where LAPSE loses most in this test. Section 12.11 looks at why.

Table 5 breaks the results down by the number of cuts. With one cut, forced LAPSE passes 22 of 25. With three or four cuts, alass is ahead, passing all six films against four for LAPSE. The groups with more cuts are small, and the ranking there could change with a handful of films.

Cuts	Films	LAPSE auto	LAPSE forced	alass	ffsubsync split
1	25	17	22	18	13
2	8	0 (+4)	6 (+2)	4 (+1)	4 (+1)
3	5	2 (+1)	4	5	1
4	1	0	0 (+1)	1	0

Table 5. Split test by number of cuts: passes, with partial results in brackets.

12.9 Time

Figure 4 shows every normal run on a logarithmic axis. Without the cache, LAPSE took a median of 10.82 s, alass 11.34 s and ffsubsync 17.25 s. LAPSE's first run was faster than alass on 26 of 38 films, with a median ratio of 0.79. With the cache, the median dropped to 0.66 s (interquartile range 0.48 to 0.88 s), 16.7 times faster than the first run and 16.3 times faster than alass. The total over 39 films was 511.2 s for the first runs and 29.7 s cached, against 514.1 s for alass and 656.3 s for ffsubsync.

In the split test all LAPSE runs were cached. The medians were 0.72 s for auto and 0.62 s for forced splitting, against 12.15 s for alass and 20.94 s for ffsubsync's split mode. The split recursion adds little to a run; the audio is what costs time.

12.10 The verdict

Table 6 crosses the verdict with the outcome. In the normal test, all 22 *solid* answers were right (95% interval for the precision 85 to 100 %). The two wrong answers, both *Godfather* films, were marked

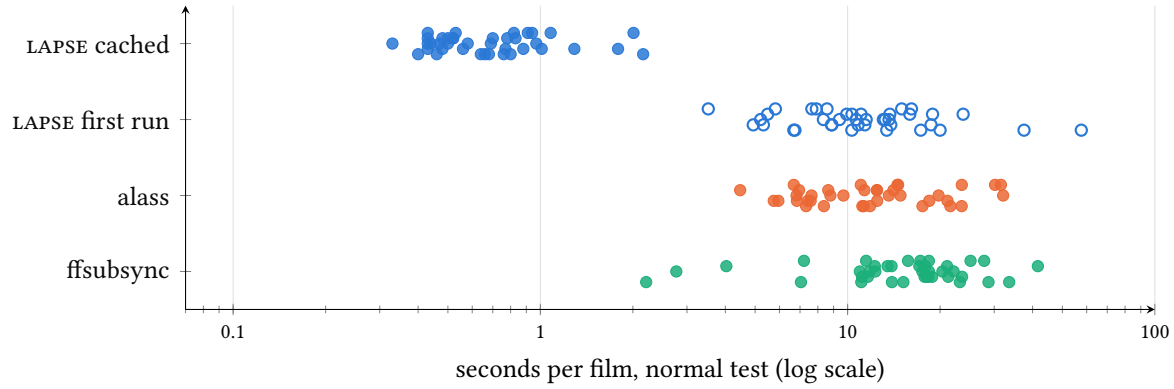

Figure 4. Time per film in the normal test, one dot per film. alass's crash on *Come and See* is left out. The four ffsubsync runs near 2 to 7 s are the `.mkv` films, where `benchmarks.md` notes the container changes its timing.

	Normal			Split, auto			Split, forced		
Verdict	pass	part	fail	pass	part	fail	pass	part	fail
solid	22	0	0	7	3	6	15	0	1
unsure	2	0	2	1	1	4	6	3	2
nothing	1	0	0	1	1	3	0	0	0

Table 6. Verdict against outcome for the 27 films of the second round.

unsure, so their originals were left alone. The cost was three correct answers held back: two *unsure* and one *nothing*, 3 of the 25 correct answers in this round. That is the trade the verdict was designed for.

With splitting forced, 15 of 16 *solid* answers passed (72 to 99 %). The one exception, *Manchester by the Sea*, was marked *solid* and failed, while the same film in the normal test passed with a verdict of *nothing*. The forced split verdict still judges the global offset (Section 10), and for this film the global offset was right while the split was wrong.

The auto column is where the verdict breaks down. Of 16 *solid* answers, 7 passed, 3 were partial and 6 failed (23 to 67 %). The cause is the first property stated after Algorithm 11: in the shifted path the verdict is reached before the split search runs, and the split result is never judged. When one segment dominates the file, the global offset is right for that segment, the verdict says so, and the split search that follows either finds nothing within its window or finds the wrong thing.

12.11 Why automatic splitting fails

The second property after Algorithm 11 predicts which films should fail: the fine split search only looks ±45 s around the global offset, and Definition 9.6 rejects any piece further than 60 s from it. For each film of the second round, take the *spread* as the difference between the largest and smallest injected segment offset. Figure 5 plots the spread against the outcome in auto mode.

Of the 6 films with a spread of 45 s or less, 4 passed and 2 failed. Of the 21 with a larger spread, 5 passed, 5 were partial and 11 failed. The benchmark did not record which path each run took; the passes above 45 s are most likely runs where the slices fell into separate groups and the recut path, with its ±600 s window, ran. This is a reading of one benchmark after the fact, with small groups, and the spread is not the only thing that changed between films. Still, the pattern matches the mechanism, and it points at a specific change: judge the split result on its own, and let the recut path run when the fine search finds nothing.

12.12 Millisecond error

The first round measured, for twelve films, how far each cue ended up from where it was in the original download (Table 13). LAPSE landed within 100 ms on five films and alass on four; ffsubsync reported an

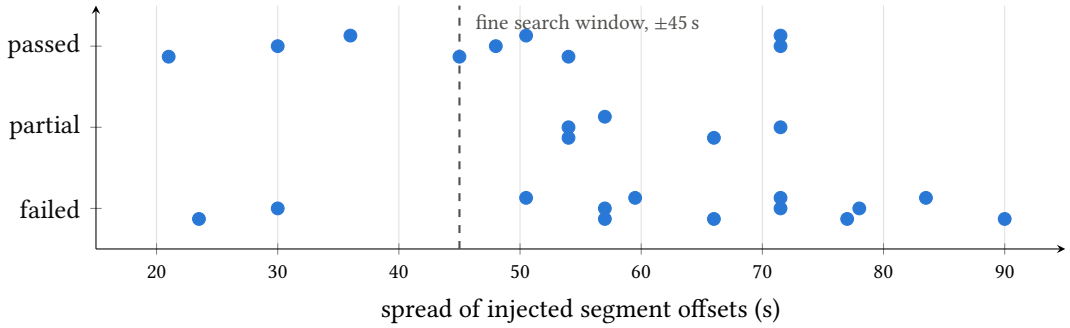

Figure 5. Outcome of LAPSE in auto mode against the spread of the injected offsets, 27 films of the second round.

error of exactly 0 ms on four, one of them a failure. The large errors that still count as passes, such as *Dancer in the Dark* at several minutes for all three tools, come from downloads that were already out of sync with the release before anything was injected: the tool moved the subtitle onto the audio, and the measure compared it with a subtitle that was not. The pass mark is the better measure for that reason, and the error table is best read film by film.

12.13 A subtitle for the wrong film

A subtitle for a documentary about Oppenheimer was run against the film *Oppenheimer*. LAPSE took 83.3 s, reported $\Lambda = 2.14$ and the verdict *nothing*, and left the original alone, with its guess in a sidecar. alass took 25.0 s, applied a shift of −6 minutes 53 s with a framerate change of 25/23.976, and wrote the file. ffsubsync took 29.0 s, applied +60 s with a ratio of 1.042, printed a score of −93 455 and an unsuccessful exit, and wrote the file anyway. ffsubsync can skip writing with `--skip-sync-on-low-quality`, which is off by default.

12.14 Failures film by film

The Godfather and Part II. Failed the normal test for all three tools. LAPSE marked both *unsure* and kept the originals. Part II also failed both split runs.

The Godfather Part III. Passed only with LAPSE, but marked *unsure*, so the correct answer went to a sidecar.

Stalker. Failed the normal test for LAPSE and ffsubsync and passed for alass. In the split test LAPSE passed in both modes. The film belongs to the first round, so no verdict was recorded.

Enter the Void. Three cuts; failed both LAPSE split runs, passed with alass.

Manchester by the Sea. One cut with a spread of 23.5 s; failed both LAPSE split runs and alass, passed with ffsubsync's split mode.

12.15 What the benchmark does not show

The benchmark compares tools. It does not isolate which parts of LAPSE carry its results. That needs ablations: the same films run with one design choice removed at a time, scored against the full engine. The variants that would answer the open questions of this report are:

- the exact sweep alone, without the FFT search;
- the FFT search without the running mean removed (Section 6.4);
- the block channel alone, without onsets;
- candidates ranked by the overlap score S instead of by Λ;
- Λ without its slice term;

- libfvad in place of Silero, and the downmix in place of the channel choice;
- a fixed penalty of 7 in place of $\max(3, \ln N)$;
- in auto mode, the verdict computed after the split instead of before it.

Some evidence exists already from development. The chunked regression was about 67 s off on a film converted between frame rates before the ratio search replaced it (Section 8), choosing a ratio by confidence picked the widest smear, and the notes in the source name *The Godfather*, *Saving Private Ryan* and *The Tree of Life* as the films the trend removal was written for. The worked example of Section 11 shows the lock statistic separating the answer from its nearest rival by a factor of four where the overlap score separates them by 10%. None of this replaces a controlled measurement, and the ablations have not been run.

13 Development history

The design history boxes in the preceding sections explain how each component reached its present form. This section puts those changes on one timeline and draws out what they had in common.

Date (2026)	Commit	Version	Change
3 July	`f0a539c`		RMS loudness per 10 ms, brute force search over ±120 s
6 July	`b3e4c25`		libfvad, one mode, binary; telephone band energy abandoned
8 July	`3136c1e`		chunked drift estimate with weighted regression
13 July	`5b579e4`		FFT cross correlation per chunk; 8 kHz by decimation
23 July	`fc7d666`		subtitle rewriting moved from Python into the engine
31 July	`25158c3`		project renamed from *subsnap* to LAPSE
4 August	`28103c1`		speech spans, overlap score and split recursion
5 August	`9a80d56`		exact single offset sweep becomes the default mode
6 August	`b3bab6d`		Silero; channel choice; resampling, accumulator, grid and regression fixes; ratio search; embedded tracks as reference
10 August	`aa975e8`	2.0.0	two channel FFT search, lock statistic, auto mode, verdicts, fixed pieces, cache
11 August	`3151a67`		benchmark published
19 to 22 August	`69b3258` …		eight more subtitle formats; bitmap tracks as reference
9 September	`ee7dadf`	2.1.0	snapping to picture cuts
21 September	`8eb820b`		batch mode
24 September	`c76f550`	2.2.3	version described in this report

Table 7. Timeline of the changes that affect synchronization. Commits that only touch packaging, the web interface or documentation are left out.

The first five weeks produced most of the ideas and most of the mistakes, and the mistakes had a common shape. They were silent. The integer accumulators returned an offset of zero with no error. The decimating resampler produced a timeline 9% too long that still correlated with something. The chunked drift search printed a slope and an intercept for every film while its wide search never ran. In each case the program ran to completion and printed plausible numbers, and in each case the failure was found only by checking a result against a film where the right answer was known. The accuracy suite in the repository came out of this: every case in it is one that broke at some point, built from synthesised speech at known times, so a failure means a fault has come back. The same experience is behind the verdict. An engine that can be wrong without noticing should at least measure how sure it is.

A second pattern was the gap between the score being maximised and the question being asked. The overlap score measures how much of the subtitle sits on speech. That is correlated with correct alignment, and it is also correlated with the amount of talking in the overlap, which has nothing to do

with it. The mean removal in August, the running mean in 2.0.0 and the lock statistic are three steps towards asking the question directly: does this offset line the file up better than chance would?

Speed changed what was affordable. The first search took minutes; after the FFT it took seconds, and that made it possible to run the search eight more times on slices, eleven times for ratios, and 96 times for a baseline. Sampling the film to save decode time was dropped for the same reason once decoding ran in parallel. Most of the accuracy of 2.0.0 is bought with time that the FFT freed up.

Some choices were made against features. The engine does not recognise words, which would give exact anchors but tie it to a language. It does not prefer the current timing of a file beyond including offset zero among its candidates, so a file that is already right can still be moved when another offset wins; a field report of exactly that is discussed in Section 14.5. It never refuses outright: a result it cannot back is still written, to a sidecar, so that the user can compare the two versions while watching and keep whichever is right. And it does not try to decide which of two subtitle files is better, only where each belongs.

14 Threats to validity and limitations

14.1 Internal validity

Tuning on the test set. The blur widths, the channel weight 1.15, the weights and clips in (28), τ, the thresholds 3.5 and 8, and the 400 ms agreement window were all adjusted while looking at these 39 films and the failures on them. The benchmark therefore overstates how LAPSE does on films it has not seen, by an amount nobody knows. alass and ffsubsync were not tuned on these films. A held out set is the fix and does not exist yet.

Who ran it. The benchmark was designed, run and scored by the author of one of the tools. The pass criterion was applied by judgement per film and not recorded as a number. A second person scoring the same outputs blind would make the counts more credible.

Embedded tracks. Four films were `.mkv` files with embedded subtitles. By default LAPSE 2.0.0 uses an embedded text track as its reference when one is present (Section 4.2), which would make those four runs subtitle to subtitle alignments instead of subtitle to audio. `benchmarks.md` does not say whether `--no-embedded` was passed. If it was not, those four results measure a different and easier task for LAPSE than for the other tools.

Configuration of the other tools. The penalty given to ffsubsync's split mode is not recorded. alass and ffsubsync ran with their defaults, which is fair to a user and not necessarily the best either tool can do.

Single runs. Each timing is one run on one machine. The differences between tools are large enough that this matters little for the medians, but individual times in Table 9 would move between runs.

14.2 External validity

Hard films, one subtitle each. The films were chosen to be difficult, so the pass rates are lower than on a typical library, as they should be. They are also chosen by one person from one library, mostly English language films with a handful in other languages, and each film has one subtitle, in SubRip. Results on television series, on animation, and on other formats are not covered.

Synthetic errors. The shifts, stretches and cuts were injected into subtitles that otherwise matched the release. A recut in the wild removes or adds footage, so a split there has cues with no speech at all on one side of the cut, and a framerate conversion outside the benchmark may come with a different edit. The synthetic split test is easier than the problem it stands for.

Versions. The benchmark measures LAPSE 2.0.0. The searches and the verdict are the same in 2.2.3 (Section 4.8), but the benchmark has not been rerun, and `benchmarks.md` itself says that later versions are expected to do better. That expectation has not been measured.

14.3 Construct validity

A pass means every cue lands close to where it belongs. It does not say how close, and the millisecond error, the one continuous measure, is taken against the original download, which is itself sometimes out of sync (Section 12.12). A partial result counts as a failure in the tests. The verdict is judged against the pass mark, so any noise in the pass mark also blurs the calibration in Table 6.

14.4 Statistical conclusion validity

With 39 films and paired binary outcomes, a test needs a lopsided split of discordant films to reach significance. Only two of the six comparisons in Section 12 do. The others are consistent with LAPSE being ahead and also with chance. The verdict calibration rests on 27 films, and the interval for the precision of *solid* in the normal test still reaches down to 85%.

14.5 Limitations of the engine

Automatic splitting. The auto mode misses cuts wider than its fine search, and its verdict does not judge the split (Section 12.11). This is the largest gap between what the engine can do when told and what it does by itself.

Moving a file that was right. Offset zero is always a candidate, but it gets no advantage beyond that. A field report on the Jellyfin plugin describes an already correct Croatian subtitle moved by −1 206 ms with a *solid* verdict. Requiring any non-zero offset to beat zero by a margin would prevent that case and would also hold back small correct shifts, which are common. The engine currently accepts the risk and relies on the backup to make a bad correction cheap to undo.

Films that defeat every tool. The first two *Godfather* films failed for all three tools. LAPSE flags them as unsure; it does not fix them.

The random baseline. The 96 random offsets are drawn from ±15 minutes, so about one run in ten has a draw within a second of the true offset. That inflates the baseline slightly and makes the verdict a little more cautious, which is the safe direction.

Drift off the list. Drift outside the eleven ratios of Table 2 is only corrected in the explicit `o1s` mode, and timing that is not linear, such as a subtitle made from a variable frame rate source, is not handled at all.

Detector warm up. Silero starts cold eight times per ten minute piece (Section 5.3). A line that starts in the first windows of a lane is detected late or not at all. With 96 lane starts in a two hour film this touches a small number of lines, and it has not been measured.

15 Conclusion and future work

LAPSE aligns subtitles to speech with an overlap score on weighted spans, an FFT search on speech blocks and onsets with the slow trend removed, a lock statistic that compares every answer with random offsets on the same file, and a decision procedure that picks the kind of correction per file. On a benchmark of 39 hard films, version 2.0.0 passed more films than alass and ffsubsync in the shift and drift test and with splitting forced on, and in the shift and drift test it marked no wrong answer as solid. The benchmark is small, and only some of the differences are clear of chance.

The next steps follow from the weak points. The ablations listed in Section 12.15 need to be run, and the thresholds need a held out set of films. The automatic split mode needs a verdict of its own,

computed after the split, and a way to reach cuts wider than 45 s without first falling into the recut path. The lock statistic could be turned into a calibrated probability by fitting it to outcomes on a larger set. And the benchmark should be repeated on 2.2.3, with the tolerance written down, the embedded tracks switched off, and a second person scoring the results.

The source code and the benchmark are at https://github.com/Schwponaco-org/lapse, under the GNU General Public License, version 3.

References

[1] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[2] Brad Jackson, Jeffrey D. Scargle, et al. An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters*, 12(2):105–108, 2005.

[3] Kaegi. Language-agnostic subtitle synchronization. Bachelor thesis, September 2019. Reference implementation: https://github.com/kaegi/alass.

[4] Rebecca Killick, Paul Fearnhead, and Idris A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[5] Stephen Macke. ffsubsync: Automagically synchronize subtitles with video. https://github.com/smacke/ffsubsync, 2019. Version 0.5.1 used here.

[6] Daniel Pirch. libfvad: Voice activity detection library, a fork of the WebRTC VAD. https://github.com/dpirch/libfvad.

[7] Peter J. Rousseeuw and Christophe Croux. Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88(424):1273–1283, 1993.

[8] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[9] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[10] Otto Seiskari. autosubsync: Automatic subtitle synchronization tool. https://github.com/oseiskar/autosubsync.

[11] Pranab Kumar Sen. Estimates of the regression coefficient based on Kendall's tau. *Journal of the American Statistical Association*, 63(324):1379–1389, 1968.

[12] Silero Team. Silero VAD: pre-trained enterprise-grade voice activity detector. https://github.com/snakers4/silero-vad.

[13] Michał Szymaniak. subsync: Subtitle speech synchronizer. https://github.com/sc0ty/subsync.

[14] Henri Theil. A rank-invariant method of linear and polynomial regression analysis. *Indagationes Mathematicae*, 12:85–91, 1950.

A Constants

Stage	Constant	Value
Input	minimum cues; cue length cap for scoring	10; 10 s
Input	MicroDVD rate from running times: tolerance	8%
Detection	sample rate; piece length; priming	16 kHz; 600 s; 3 s
Detection	channel probe	180 s from 35%
Detection	plausible share of speech	0.15 to 0.65
Detection	Silero window; context; lanes	512; 64 samples; 8
Detection	Silero gain	$0.1/L_{90}$, clipped to 1 to 12
Detection	libfvad gain	$0.05/L$, clipped to 0.5 to 40; $\alpha = 0.005$
Detection	music filter	±50 frames; CV below 0.35
Spans	speech threshold; merge gap; minimum length	0.25; 200 ms; 200 ms
FFT search	bin; lag range	40 ms; ±15 min
FFT search	running mean, blocks; onsets	30 s; 8 s
FFT search	onset kernel; onset weight	120 ms; 1.15
FFT search	blur per pass	0, 550, 450, 250 ms
FFT search	peaks per pass; exclusion; pooled spacing; pool size	10; ±2.5 s; 2 s; 40
Refinement	candidates refined; coarse scan; fine scan	4; ±1.2 s at 5 ms; ±6 ms at 1 ms
Lock	τ; whole file draws; slice draws; slices	450 ms; 96; 48; 8
Lock	weights; slice clip	0.6, 1.6; $[-1, 3]$
Lock	random offset range; seeds	±900 s; 12345, 777
Sweep path	bucket; excluded neighbours	1 s; ±2
Drift	ratio snap tolerance; minimum slope	0.004; 10^{-5}
Drift	regression chunks; first chunk; minimum sharpness	15 min; 7.5 min; 1.05
Modes	slice agreement η; slices for shift; for line	400 ms; 3; 4
Modes	stretch must beat shift by; ratio search must beat by	1; 2
Split	recut grid	±600 s at 100 ms
Split	fine grid inside auto	±45 s at 25 ms
Split	penalty; boundary reach	$\max(3, \ln N)$; 60 cues
Split	worth splitting: run length; distance; range	12 cues; 60 s; 120 ms
Restart	backward jump; part search	60 s; ±90 min
Joined	slices; jump; confidence floor	16; 5 min; 0.02
Verdict	nothing below; solid from; agreeing slices	3.5; 8; 3
Snap	window; minimum median keyframe gap; fixed GOP test	120 ms; 400 ms; ±100 ms
Cache	minimum spans	20

B Per film results

All tables in this appendix reproduce the per film rows of `docs/benchmarks.md`. ✓ is a pass, ✗ a failure and ◦ a partial result with the share of segments that landed.

Table 9. Normal test, 39 films. Times in seconds. The LAPSE result is the same for the first and the cached run.

Film	LAPSE 2.0.0			alass 2.0.0		ffsubsync 0.5.1	
	first	cached		time		time	
2001: A Space Odyssey	13.41	0.68	✓	21.60	✓	2.21	✗
Andrei Rublev	5.32	0.43	✓	6.84	✓	11.64	✓
Apocalypse Now	9.41	0.97	✓	14.86	✓	22.13	✓
Badlands	9.93	0.48	✓	14.10	✓	13.90	✓
Barry Lyndon	16.14	0.91	✓	30.19	✓	27.83	✓
Blue Velvet	6.74	0.46	✓	8.37	✓	13.94	✓

Film	first	cached		alass		ffsubsync	
Brazil	8.86	0.88	✓	12.48	✓	18.84	✓
Challengers	13.64	0.58	✓	19.78	✓	2.77	✓
Come and See	15.94	0.70	✓	0.09 (crash)	✗	4.03	✗
Dancer in the Dark	3.52	0.43	✓	4.47	✓	7.21	✓
Drive	6.68	0.40	✓	6.68	✓	11.08	✓
Enter the Void	8.89	0.77	✓	11.29	✓	18.07	✓
Eraserhead	10.67	0.33	✓	5.75	✗	10.96	✓
Ex Machina	5.49	0.50	✓	7.63	✓	12.26	✓
Eyes Wide Shut	7.65	0.82	✓	11.34	✓	17.25	✓
Fallen Angels	10.32	0.64	✓	14.60	✓	15.19	✓
Fargo	4.93	0.56	✓	7.33	✓	11.14	✓
Hereditary	13.25	0.43	✓	18.44	✓	17.51	✓
Inland Empire	18.90	0.78	✓	32.08	✓	41.62	✓
Manchester by the Sea	7.91	1.08	✓	8.65	✗	15.73	✓
Mulholland Drive	20.00	0.80	✓	23.53	✓	23.20	✓
No Country for Old Men	11.38	0.48	✓	17.53	✓	17.81	✓
Possession	8.34	0.50	✓	5.95	✓	11.78	✓
Requiem for a Dream	13.72	0.52	✓	8.79	✗	17.89	✓
Saving Private Ryan	8.57	0.94	✓	12.46	✓	18.39	✓
Seven Samurai	37.53	0.76	✓	11.04	✓	7.05	✓
Stalker	10.82	1.79	✗	11.17	✓	18.32	✗
Synecdoche, New York	13.07	0.69	✓	21.14	✓	18.40	✓
The Deer Hunter	11.06	0.83	✓	13.62	✓	21.08	✓
The Godfather	15.00	2.01	✗	12.46	✗	25.13	✗
The Godfather Part II	17.30	2.16	✗	14.53	✗	28.75	✗
The Godfather Part III	13.82	1.29	✓	11.83	✗	23.58	✗
The Hunt	11.49	0.44	✓	7.42	✓	20.35	✓
The Passenger	23.77	0.43	✓	9.69	✓	17.12	✓
The Seventh Seal	5.82	0.53	✓	6.95	✓	11.49	✓
There Will Be Blood	57.63	0.66	✓	31.55	✓	33.57	✓
Uncut Gems	18.69	1.01	✓	23.51	✓	21.25	✓
Whiplash	5.21	0.47	✓	7.58	✓	12.31	✓
Zoolander	10.33	0.52	✓	6.82	✗	13.49	✓

Table 10. Split test, 39 films. All LAPSE runs cached.

Film	Cuts	LAPSE auto	LAPSE forced	alass	ffsubsync	ffsubsync split
2001: A Space Odyssey	1	✓	✓	✓	✗	✗
Andrei Rublev	1	✓	✓	✓	✗	✓
Apocalypse Now	2	◦ 2/3	✓	✓	✗	✓
Badlands	3	◦ 2/3	✓	✓	✗	✗
Barry Lyndon	3	✓	✓	✓	◦ 2/3	✓
Blue Velvet	1	✓	✓	✓	✗	✓
Brazil	2	◦ 2/3	✓	✓	✗	✓
Challengers	1	✓	✓	✓	✗	✓
Come and See	1	✗	✓	✗	✗	✗
Dancer in the Dark	1	✓	✓	✓	✗	✗
Drive	2	◦ 2/3	✓	✓	✗	✗
Enter the Void	3	✗	✗	✓	✗	✗
Eraserhead	2	✗	◦ 2/3	◦ 2/3	✗	✗
Ex Machina	2	✗	✓	✗	◦ 2/3	✓
Eyes Wide Shut	1	✓	✓	✓	✗	✓
Fallen Angels	1	✗	✓	✗	✗	✗
Fargo	1	✓	✓	✓	✗	✓
Hereditary	1	✓	✓	✓	✗	✓

Film	Cuts	LAPSE auto	LAPSE forced	alass	ffsubsync	ffsubsync split
Inland Empire	3	✓	✓	✓	✗	✗
Manchester by the Sea	1	✗	✗	✗	✗	✓
Mulholland Drive	1	✓	✓	✓	✗	✗
No Country for Old Men	1	✓	✓	✓	✗	✓
Possession	1	✗	✓	✓	✗	✗
Requiem for a Dream	1	✓	✓	✗	✗	✗
Saving Private Ryan	1	✗	✓	✓	✗	✓
Seven Samurai	1	✓	✗	✗	✗	✗
Stalker	1	✓	✓	✓	✗	✗
Synecdoche, New York	3	✗	✓	✓	✗	✗
The Deer Hunter	2	◦ 2/3	◦ 2/3	✓	✗	◦ 2/3
The Godfather	1	✗	✓	✗	✗	✗
The Godfather Part II	1	✗	✗	✗	✗	✗
The Godfather Part III	2	✗	✓	✗	✗	✗
The Hunt	4	✗	◦ 3/4	✓	✗	✗
The Passenger	1	✓	✓	✓	✗	✓
The Seventh Seal	1	✓	✓	✓	✓	✓
There Will Be Blood	1	✗	✓	✓	✗	✓
Uncut Gems	1	✓	✓	✓	✗	✗
Whiplash	1	✓	✓	✓	✗	✓
Zoolander	2	✗	✓	✗	✗	✓

Table 11. Cuts injected in the second round. Offsets per segment in seconds.

Film	Cuts	Cut at	Segment offsets
Apocalypse Now	2	00:59:23	+33.0, -21.0, +33.0
Badlands	3	00:25:07, 00:40:52	-21.0, +45.0, +12.0, -21.0
Barry Lyndon	3	00:44:48, 01:22:52, 02:13:49	-33.0, +21.0, -33.0, +12.0
Blue Velvet	1	00:55:20	+33.0, -38.5
Brazil	2	00:38:54, 01:41:59	+38.5, -33.0, +21.0
Drive	2	00:29:49, 01:09:21	+21.0, -33.0, +21.0
Enter the Void	3	00:31:50, 01:23:52, 01:45:36	-12.0, -45.0, -15.0, +33.0
Eraserhead	2	00:26:30, 00:58:41	-15.0, +15.0, -12.0
Ex Machina	2	00:29:25, 01:02:30	+21.0, +38.5, -12.0
Eyes Wide Shut	1	01:10:21	+12.0, +33.0
Fallen Angels	1	00:43:05	-45.0, +45.0
Fargo	1	00:44:28	-45.0, -15.0
Hereditary	1	00:56:00	+15.0, -21.0
Inland Empire	3	00:51:53, 01:31:06, 02:11:54	-12.0, +12.0, -33.0, +12.0
Manchester by the Sea	1	00:58:58	+15.0, +38.5
Mulholland Drive	1	01:14:58	+38.5, -12.0
No Country for Old Men	1	01:02:25	+33.0, -15.0
Possession	1	00:53:52	-33.0, +33.0
Synecdoche, New York	3	00:38:13, 01:04:41, 01:32:22	-38.5, -15.0, +21.0, +38.5
The Deer Hunter	2	00:57:45, 01:51:49	-12.0, +21.0, +45.0
The Godfather	1	01:32:14	-45.0, +12.0
The Godfather Part II	1	01:31:22	+38.5, -21.0
The Godfather Part III	2	00:56:24, 01:43:55	-15.0, -33.0, +38.5
The Hunt	4	00:22:03, 00:41:17, 01:11:23, 01:27:20	-15.0, +15.0, +38.5, -15.0, -33.0
The Passenger	1	01:03:37	+33.0, -38.5
There Will Be Blood	1	00:36:16	-45.0, +12.0
Zoolander	2	00:26:30, 00:59:43	+12.0, +38.5, -45.0

Table 12. LAPSE verdicts and outcomes, second round.

Film	Normal		Split, auto		Split, forced	
Apocalypse Now	solid	✓	unsure	◦	unsure	✓
Badlands	solid	✓	solid	◦	solid	✓
Barry Lyndon	solid	✓	solid	✓	solid	✓
Blue Velvet	solid	✓	nothing	✓	solid	✓
Brazil	solid	✓	nothing	◦	solid	✓
Drive	solid	✓	solid	◦	solid	✓
Enter the Void	solid	✓	solid	✗	unsure	✗
Eraserhead	solid	✓	solid	✗	unsure	◦
Ex Machina	solid	✓	solid	✗	solid	✓
Eyes Wide Shut	solid	✓	solid	✓	solid	✓
Fallen Angels	unsure	✓	unsure	✗	unsure	✓
Fargo	solid	✓	solid	✓	solid	✓
Hereditary	solid	✓	solid	✓	solid	✓
Inland Empire	solid	✓	solid	✓	solid	✓
Manchester by the Sea	nothing	✓	nothing	✗	solid	✗
Mulholland Drive	solid	✓	solid	✓	solid	✓
No Country for Old Men	solid	✓	solid	✓	solid	✓
Possession	solid	✓	solid	✗	unsure	✓
Synecdoche, New York	solid	✓	solid	✗	solid	✓
The Deer Hunter	solid	✓	solid	◦	unsure	◦
The Godfather	unsure	✗	unsure	✗	unsure	✓
The Godfather Part II	unsure	✗	unsure	✗	unsure	✗
The Godfather Part III	unsure	✓	unsure	✗	unsure	✓
The Hunt	solid	✓	nothing	✗	unsure	◦
The Passenger	solid	✓	unsure	✓	solid	✓
There Will Be Blood	solid	✓	solid	✗	solid	✓
Zoolander	solid	✓	nothing	✗	unsure	✓

Table 13. Error in milliseconds against the original download, first round, normal test.

Film	Injected	LAPSE	alass	ffsubsync
2001: A Space Odyssey	shift +97000 ms	19 ✓	50 ✓	34 769 ✗
Andrei Rublev	shift +15000 ms	35 ✓	168 ✓	0 ✓
Challengers	ratio 1.00100	67 ✓	46 ✓	160 ✓
Come and See	shift +97000 ms	24 082 ✓	crash ✗	33 110 ✗
Dancer in the Dark	ratio 0.99900	384 035 ✓	304 591 ✓	195 928 ✓
Requiem for a Dream	shift −33000 ms	2 113 ✓	140 672 ✗	1 790 ✓
Saving Private Ryan	ratio 1.00100	3 646 ✓	3 574 ✓	3 525 ✓
Seven Samurai	ratio 1.04167	60 ✓	38 ✓	0 ✓
Stalker	shift +52500 ms	89 302 ✗	101 286 ✓	0 ✗
The Seventh Seal	shift +33000 ms	58 ✓	26 ✓	0 ✓
Uncut Gems	shift −33000 ms	16 076 ✓	16 105 ✓	16 054 ✓
Whiplash	shift +38500 ms	266 ✓	172 ✓	170 ✓

C Running the engine

The engine is built and run from the repository:

```
cmake -B build && cmake --build build -j       # the engine
./build/lapse video.mkv subtitles.srt           # auto mode
./build/lapse video.mkv subtitles.srt split     # forced split
./build/lapse video.mkv subtitles.srt --json    # machine readable result
```

The benchmark films are not distributed. The error injection is described in Section 12, and the injected cuts of the second round are listed in Table 11.