LAPSE



Language-Agnostic Playback Synchronization Engine

Technical report, version 2.5.0

Rasmus Stisen Jensen

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

November 2026

What this edition covers.

Engine	LAPSE 2.5.0, November 2026.
Measurements	LAPSE 2.5.0 and 2.0.0 against alass 2.0.0 and ffsubsync 0.5.1 on 39 feature films, published in `docs/benchmarks.md` on 13 November 2026 (commit `9ef92ea`). Every film was run again on 2.5.0. alass and ffsubsync keep their numbers from the August round and were not run again.
Worked example	Run on 2.5.0 with extra logging, to show what the engine computes. It is an illustration and is not used as evidence of accuracy.
Earlier edition	September 2026, describing 2.2.3 with measurements of 2.0.0. This edition replaces it.

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, advert breaks and files that hold two parts, and it does so without reading the language of either side. This edition describes version 2.5.0. Speech is detected with Silero VAD, or with libfvad when the neural model is missing, and turned into weighted spans that start 150 ms before the detector fires. Offsets are found with a two channel FFT cross correlation on speech blocks and speech onsets, with a running mean removed from both channels. Under a doubly stochastic point process model of speech onsets, that removal keeps 95 to 96% of the energy of the peak at the correct offset and a fraction $\frac{47}{10080}(W/\tau)^3$ of the slow component that dense dialogue adds, with W the width of the running mean and τ the correlation time of the dialogue density. Candidates are refined on an exact overlap score. That score is a sum of truncated distances to plateaus, so its local maximiser is a weighted interval median, and its asymptotic efficiency under Gaussian timing noise is $2/\pi$. Candidates are ranked by a lock statistic that compares each offset with the same file at random offsets. Drift is searched over the standard framerate pairs and then, if none fits, by a grid and golden section search over ratios between 0.94 and 1.06. Files that need splitting are aligned by penalised dynamic programming on an irregular grid of offsets, with an unconstrained jump at twice the penalty for scenes that are missing from the video. The verdict that decides whether the original file is overwritten is computed on the file that the split leaves behind.

On 39 feature films chosen for being hard to synchronise, LAPSE 2.5.0 passed 37 in the shift and drift test, against 31 for alass 2.0.0 and 33 for ffsubsync 0.5.1. With cuts injected it passed 34 when it had to find the cuts by itself, up from 19 for LAPSE 2.0.0 (exact McNemar $p = 6.1 \times 10^{-5}$), and 37 with splitting forced on, against 28 for alass and 18 for ffsubsync's split mode. On the 27 films where verdicts were recorded, all 22, 16 and 17 of its *solid* verdicts in the three tests were correct. A first run on all 39 films took 373 s, 27% less than 2.0.0. A run from the speech cache is about a fifth slower than in 2.0.0, because a wide search for distant parts of the file runs every time.

Contents

1 Introduction **4**
1.1 The problem . . . 4
1.2 What makes it hard . . . 4
1.3 Approach . . . 4
1.4 What changed since the previous edition . . . 4
1.5 Contributions . . . 5
1.6 Scope and layout . . . 5

2 Background and related work **5**
2.1 How subtitles go out of sync . . . 5
2.2 Span alignment: alass . . . 6
2.3 Binary strings and FFTs: ffsubsync . . . 6
2.4 Recognition, learned detectors and audio to audio . . . 6
2.5 Voice activity detection . . . 6
2.6 Time delay estimation . . . 7
2.7 Changepoints and warping . . . 7
2.8 Robust location and slope . . . 7

3 Formal model **7**
3.1 Spans and sequences . . . 7
3.2 Error models . . . 8
3.3 The pair score . . . 8
3.4 Objectives . . . 9
3.5 How many cues are enough . . . 9

4 Architecture and libraries **10**
4.1 The pipeline . . . 10
4.2 Where the reference comes from . . . 10
4.3 Libraries . . . 10
4.4 Determinism . . . 12
4.5 Subtitle preprocessing . . . 12
4.6 Writing the result . . . 13
4.7 The speech cache . . . 13
4.8 Changes between 2.2.3 and 2.5.0 . . . 14

5 Speech detection **14**
5.1 Decoding . . . 14
5.2 Choosing the channel . . . 14
5.3 Silero . . . 15
5.4 The libfvad fallback . . . 15
5.5 From probabilities to spans . . . 16
5.6 Lead-in . . . 16
5.7 Words as a filter . . . 17
5.8 How the detector got here . . . 17

6 Searching for one offset **18**
6.1 The exact sweep . . . 18
6.2 Why the maximum of S is not enough . . . 18
6.3 A point process view . . . 19
6.4 Two channels on a grid . . . 21
6.5 Removing the trend . . . 22
6.6 Four passes with different blur . . . 23
6.7 Transform length and wrap around . . . 23
6.8 Peaks and their height . . . 24

7 Refinement and the lock statistic **24**
7.1 Refinement on the exact score . . . 24
7.2 What the refinement estimates . . . 25
7.3 Bunching . . . 26
7.4 A baseline from random offsets . . . 26
7.5 Slices, and the statistic itself . . . 27
7.6 The complete search . . . 28
7.7 The statistic on the sweep path . . . 28

8 Drift **29**
8.1 The ratios that occur . . . 29
8.2 Why measuring the lag chunk by chunk fails . . . 29
8.3 Searching the ratios . . . 29
8.4 The slope from eight slices . . . 30
8.5 Ratios between the pairs . . . 30
8.6 Drift and a cut together . . . 31
8.7 The explicit mode . . . 32

9 Split alignment **32**
9.1 The grid . . . 32
9.2 Keeping cues in order . . . 33
9.3 The recursion . . . 33
9.4 Checking runs and settling boundaries . . . 34
9.5 Chapters . . . 35
9.6 The penalty . . . 36
9.7 When a split is kept . . . 36
9.8 Restarts and joined parts . . . 36

10 Choosing the correction and judging it **37**
10.1 Evidence from slices . . . 37
10.2 What agreement is worth . . . 37
10.3 Looking for a far part . . . 38
10.4 The decision . . . 38
10.5 Reaching further . . . 39
10.6 The verdict . . . 39
10.7 From one release to another . . . 40
10.8 Snapping to picture cuts . . . 41

11 A worked example **41**
11.1 Input and speech . . . 42
11.2 The shifted copy . . . 42
11.3 The drifting copy . . . 43
11.4 The cut copy . . . 44

12 Evaluation **44**
12.1 Questions . . . 44
12.2 Material . . . 45
12.3 Error injection . . . 45
12.4 Tools and configurations . . . 45
12.5 Scoring and statistics . . . 46
12.6 Shift and drift . . . 46
12.7 Cuts . . . 46
12.8 Time . . . 47
12.9 The verdict . . . 48
12.10 Spread and automatic splitting . . . 49
12.11 Millisecond error . . . 50
12.12 A subtitle for the wrong film . . . 50
12.13 Failures film by film . . . 50
12.14 What the benchmark does not show . . . 50

13 Development history **51**

14 Threats to validity and limitations **52**
14.1 Internal validity . . . 52
14.2 External validity . . . 52
14.3 Construct and statistical validity . . . 53
14.4 Limitations of the engine . . . 53

15 Conclusion and future work **54**

A Constants **56**

B Results per film **57**

C Running the engine **61**

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. Television recordings add a fourth case that is really the third one repeated: an offset that jumps at every advert break.

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 is to find where people talk in the audio and slide the the subtitle until its cues sit on top of that talking. Every tool in this area does some version of that [15, 19, 28].

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 or shouted over an engine, and some of it sits on a channel the downmix buries. Any detector will call some music speech and miss some speech, and the ones that work best also tend to fire a little after a word has started.

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 audio supports the answer turns a problem the user can see, subtitles out of sync, into one they cannot see, 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 [15]. 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 wider search looks for parts of the file that sit far from the rest. A verdict built on the lock statistic of the finished answer and on the agreement of the slices decides whether the result replaces the original or goes to a sidecar file beside it.

1.4 What changed since the previous edition

The previous edition of this report described version 2.2.3 and closed with a list of weak points. The largest was automatic splitting. In 2.0.0 the verdict was computed for the global offset before any split was tried, the fine split search could only reach 45 s from that offset, and on the benchmark the engine passed 19 split files when left to itself against 32 when told to split.

Version 2.5.0, the one described here, changes this in four places. The split is now tried for an unsure answer as well, and every answer in pieces is judged by the lock of the file once the pieces have been moved. A search over sixteen slices reaching 90 minutes runs on every file and looks for parts that sit far away. The split recursion works on an irregular grid that includes those far offsets, and it may jump past a scene missing from the video at twice the usual penalty. And when a drifting file is also cut, the framerate is chosen by how well the file fits after it has been split. Smaller changes run

through the rest of the pipeline: speech spans start 150 ms early, the drift search covers ratios outside the standard pairs, the soundtrack is decoded in three minute pieces with the mixes probed side by side, and the verdict's lower threshold moved from 3.5 to 6. Section 4.8 lists them.

On the same 39 films, the automatic split went from 19 passes to 34. That is the main result of this edition. It is also the one that most needs the caveat of Section 14, since the films are the same ones the engine was tuned on.

1.5 Contributions

The report makes six contributions. Some are mathematical statements about parts of the engine that were designed by trial, and they explain after the fact why those parts work.

1. An exact description of the overlap score as a sum of truncated distances to plateaus (Proposition 3.6). Locally, the best offset is a weighted interval median, with the efficiency of a median under Gaussian noise (Proposition 7.4).
2. A cross spectral decomposition of the expected correlation between speech onsets and cue starts under a doubly stochastic point process model (Proposition 6.4). The trend removal keeps most of the term that locates the offset and a fraction $\frac{47}{10080}(W/\tau)^3$ of the term that does not (Proposition 6.5).
3. A lock statistic with bounds and invariance properties, and a floor on its baseline spread that makes it finite on every input (Section 7).
4. A continuous framerate search with a stated resolution, and a closed form for the bias that a cut adds to the Theil and Sen slope (Proposition 8.4), which explains why a drifting file that is also cut used to get the neighbouring framerate.
5. A split recursion over an irregular grid of offsets with three kinds of transition, with a proof of optimality and a cost of $O(N|\Sigma|)$ time and $2N|\Sigma|$ bytes (Theorem 9.3).
6. A decision procedure and a verdict that judges the finished answer, and a record of how well the verdict matched the outcomes on a hard benchmark (Section 12.9).

None of these is deep on its own.

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

1.6 Scope and layout

The Docker watcher, the web interface, the Jellyfin plugin and the translation feature that ship with LAPSE are left out. So are the subtitle parsers and writers, except where they affect timing. Where this report quotes a benchmark number, it comes from the latest round of `docs/benchmarks.md`; where it quotes a number from development, it says so, because those numbers come from test material that is not published.

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, most of them ending with a short account of the designs that came before. 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 [15] 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 advert 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, and that combination turned out to need its own treatment (Section 8.6).

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 [19] 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, and a golden section search over the ratio is available behind `--gss`. 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, learned detectors and audio to audio

A second family of tools recognises words. subsync [31] runs speech recognition on the audio and matches the recognised words with the words in the subtitle. That gives precise anchors when it works, and it needs a language model for the language of the film and of the subtitle. autosubsync [28] stays language agnostic and trains a small classifier to detect speech in the audio, then searches jointly over offset and framerate. Sushi [6] solves a neighbouring problem: it moves a subtitle from one release to another by comparing the two soundtracks, which is common for fan subtitles timed to a television broadcast. LAPSE has the same feature as `--from` (Section 10.7).

LAPSE does not match words against the subtitle. 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. It can still use a recogniser in a weaker role. Whisper [23], run through whisper.cpp [10], reports when it heard words, and with `--whisper` only the stretches of detected speech that contain a word are kept (Section 5.7). The words themselves are never compared with the subtitle.

2.5 Voice activity detection

Voice activity detectors come in roughly three generations [24]. 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 [22], computes log energies in six sub bands and feeds them to Gaussian mixture models for speech and noise, with four modes of increasing agressiveness; it makes a

decision every 10, 20 or 30 ms. Neural detectors such as Silero VAD [30] run a small recurrent network on raw audio in windows of 32 ms and return a probability. They separate speech from music much better, and they are 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 Time delay estimation

Finding the offset between two signals is the classical time delay estimation problem. Knapp and Carter [18] wrote the cross correlation with a frequency weighting applied to the cross spectrum, the generalised cross correlation, and showed that the maximum likelihood weighting for two noisy copies of one signal depends on the coherence of the two at each frequency. The trend removal of Section 6.5 is a fixed weighting of this kind, and Section 6.3 gives the reason for its shape. The setting differs from the classical one in two respects. The signals are point processes and indicator functions [2, 5], and the noise that matters most is a slow, structured component that both signals share, which a stationary noise model does not describe well.

2.7 Changepoints and warping

Split alignment is a changepoint problem [33]. The offset is piecewise constant along the file, and each change is paid for with a penalty. Penalised optimal partitioning [14] solves this kind of objective exactly by dynamic programming [3], and PELT [17] prunes the search to make it linear in the typical case. A penalty that grows with $\ln n$ has the form of the Bayesian information criterion [27]. Dynamic time warping [26] is the familiar tool for aligning two time series, and it lets the alignment change at every step. A recut film jumps and then holds still, which is why LAPSE pays for every change.

2.8 Robust location and slope

Medians turn up all over the engine. 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 [25], and the Theil and Sen estimator of a slope, the median of all pairwise slopes [29, 32], which tolerates corruption of up to about 29% of the points. M-estimation [13] and the influence curve [11] give the vocabulary for Section 7.2, where the exact score turns out to behave like a median itself.

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 [15]; 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 patterns of Section 2 become four models:

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

$$\text{drift:} \quad t' = \rho\, t + \sigma, \qquad \rho \in [5/6,\ 6/5], \tag{2}$$

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

$$\text{drift and split:} \quad t' = \rho\, t + \sigma_n. \tag{4}$$

In practice ρ is one of eleven framerate ratios (Table 3), and the engine also searches the interval $[0.94,\ 1.06]$ continuously. LAPSE reports ρ as `ratio`, σ as `offset_ms`, and the offsets of a split as `offsets` with the cues where they start as `splits`.

With `--from`, the subtitle is known to fit a second video, and the engine aligns the speech of the second video against the speech of the first under the same four models; Section 10.7 describes how the result is carried back to the cues.

3.3 The pair score

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

$$\operatorname{ov}(a,b) = \max\bigl(0,\ \min(a_2,b_2) - \max(a_1,b_1)\bigr), \qquad \operatorname{iscore}(a,b) = \frac{\operatorname{ov}(a,b)}{\min(\operatorname{len} a, \operatorname{len} b)}, \tag{5}$$

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

$$\phi_{kn}(\sigma) = h_k\, w(r^k, c^n)\, \operatorname{iscore}(r^k, c^n + \sigma). \tag{6}$$

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 shorter span begins to 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 $P_{kn} = [\beta_2, \beta_3]$ is the set of offsets at which the shorter span lies inside the longer one. Its length is $L - \ell$, so it shrinks to a point when the two spans have the same length.

Proposition 3.6 (Plateau distance). *With* $\operatorname{dist}(\sigma, P) = \min_{x \in P} |\sigma - x|$,

$$\phi_{kn}(\sigma) = \frac{h_k}{L_{kn}} \left(\ell_{kn} - \operatorname{dist}(\sigma, P_{kn})\right)^+. \tag{7}$$

Proof. On the plateau the distance is zero and the value is $h\ell/L$. On the rising edge $[\beta_1, \beta_2]$, $\operatorname{dist}(\sigma, P) = \beta_2 - \sigma$ and $\ell - (\beta_2 - \sigma) = \sigma - \beta_1$, which is the overlap there. On the falling edge, $\operatorname{dist}(\sigma, P) = \sigma - \beta_3$ and $\ell - (\sigma - \beta_3) = \beta_4 - \sigma$. Outside (β_1, β_4) the distance is at least ℓ and the positive part is zero. □

The identity says that every pair votes for an interval of offsets, with a vote that falls off linearly with the distance from that interval and stops at ℓ. It is the form that makes the behaviour of the refinement step easy to read (Section 7.2).

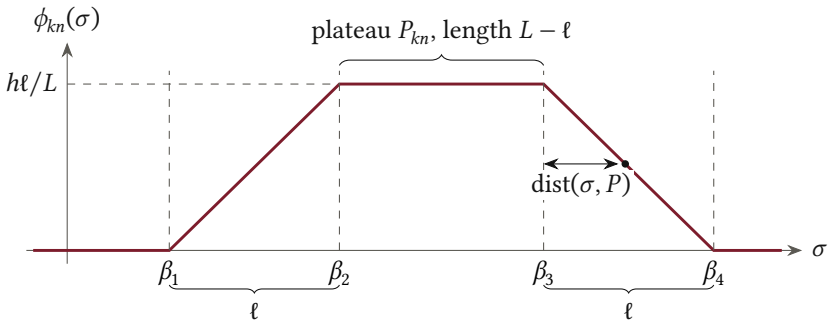


Figure 1. The pair score of Lemma 3.5. By Proposition 3.6 its height at any offset is h/L times ℓ minus the distance to the plateau.

3.4 Objectives

The single offset score is

$$S(\sigma) = \sum_{n=1}^{N} \sum_{k=1}^{K} \phi_{kn}(\sigma) = \sum_{n,k} \frac{h_k}{L_{kn}} \big(\ell_{kn} - \operatorname{dist}(\sigma, P_{kn})\big)^+. \tag{8}$$

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 $\min\big(1, S(\hat{\sigma})/N\big)$ 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

$$S(\sigma_1, \dots, \sigma_N) = \sum_{n} \sum_{k} \phi_{kn}(\sigma_n) \; - \sum_{n \geq 2} \pi_n(\sigma_{n-1}, \sigma_n), \tag{9}$$

where π_n is zero when the offset stays and otherwise a penalty that Section 9.3 defines. A cue is worth at most 1, since the reference spans do not overlap and $h_k \leq 1$, so the basic 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 more common than it sounds. Many release groups ship subtitle file that holds nothing but their own name.

4 Architecture and libraries

4.1 The pipeline

Figure 2 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.

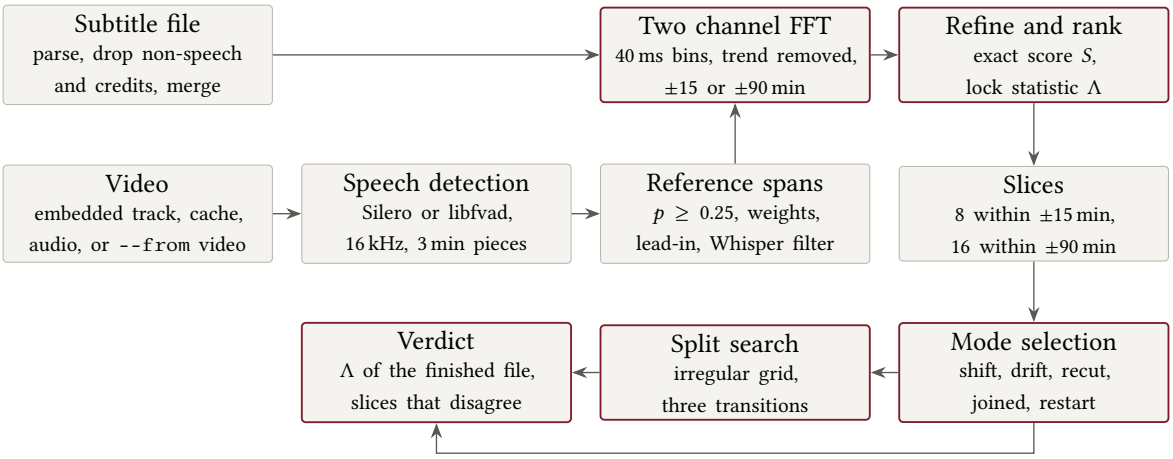


Figure 2. One run of the engine. Framed boxes are covered in Sections 6, 7, 9 and 10. A reference subtitle file skips the audio branch. The direct path from mode selection to the verdict is taken when no split is kept.

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. These timings are cached, because reading a track means reading the whole file, which on network storage takes minutes for a large film;
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 run took a median of 0.71 s against 8.82 s for the first run.

4.3 Libraries

LAPSE is a C++ program of about 6 300 lines, built on the libraries in Table 1.

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): the default downmix, the centre channel alone, or the average of front left and right. The centre or the front pair goes to the resampler on its own through a channel map, so only the channels that end up in the mix are resampled.[1] The resampler runs with a filter of 16 taps and 256 phases. The default of 32 taps and 1 024 phases is made for listening (the engine only needs to hear where words are), and on a 55

[1] An earlier version resampled all six channels of a 5.1 track and then threw five away. On a two hour AC-3 film in development, the channel map cut the CPU time of a whole sync from 86 to 73 s, with the same samples out.

Library	Role in LAPSE	Alternatives and why not
FFmpeg [8]	Demuxing every container, decoding audio, resampling to 16 kHz mono with a chosen mix, reading keyframes, chapters and embedded subtitle packets.	None realistic for this breadth of formats. GStreamer covers similar ground with a much heavier runtime.
FFTW3 [9]	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 [21]	Running the Silero model.	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 [30]	Speech probability per 32 ms window.	WebRTC is weaker on music. Larger neural detectors, such as the segmentation models in pyannote, need PyTorch, a heavy dependency for a single binary.
libfvad [22]	Fallback detector when ONNX Runtime or the model is missing.	It is the WebRTC detector with a C interface and no other dependencies.
whisper.cpp [10]	Optional, with `--whisper`: word times that decide which speech spans are kept.	Called as the external program `whisper-cli`, so the engine neither links nor ships it.

Table 1. Libraries and their roles.

minute episode the shorter filter took a cold run from 9.7 to 8.4 s, with the offset 15 ms apart and the lock statistic at 16.0 against 16.2.

When a frame arrives with a different channel count, rate or sample format, which happens when a television recording drops from 5.1 to stereo for the adverts, the resampler is set up again for it. 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. Two plans can exist per film. The first covers lags of ±15 minutes and is made when the reference is set up. The second covers ±90 minutes and is made the first time something asks that far, so a file that never needs it never pays for it (Section 6.7). The reference transforms are computed once per plan and reused for every query, and queries execute the plan on new arrays through the new array interface (`fftw_execute_dft_r2c`).

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 already runs pieces of the film in parallel (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. Linux builds are made on Ubuntu 22.04 and on Alpine, so that they start on older CPUs and in small containers. 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 180 s whatever 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 is why `--threads`, meant for servers that transcode at the same time, can cap the number of pieces in flight without changing the answer. On a two hour 5.1 film the cache and the JSON line came out identical with and without `--threads 2`; the run took 38 s of wall time and 124 MB, against 15 s and 228 MB without the cap. The proposition is about threads on one machine. 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. Sixteen file extensions are read and written, from SubRip to EBU STL [7] and Scenarist SCC, together with TTML saved as `.xml`, which is only read, and files named `.txt`, which are opened to see what they hold. 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. 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 $f/25$ times as long as it should, so

$$\hat{f} = 25 \cdot \frac{T_{\text{ours}}}{T_{\text{theirs}}}, \tag{10}$$

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 (four note signs such as ♪ and ♫, and `#`, `*`, `~`); 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. A cue is also excluded when its text contains any of nineteen credit and advert marks, such as `www.`, `synced by`, `opensubtitles` and the Danish `tekstet af`. Such a line usually sits over the opening titles, where there is no dialogue to land on. 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; flag *sort*
Output: valid spans $c^1, \dots, c^N$ and a map $\mu : \{1..M\} \to \{1..N\}$
1 $Y \leftarrow \{(x^i, i) : \operatorname{len} x^i > 0\}$, swapping start and end where reversed
2 **if** *sort* **then** sort Y by start time
3 **end if**

```
4  for (y, i) ∈ Y in order do
5      if c is not empty and c1^last ≤ y1 < c2^last then
6          c2^last ← max(c2^last, y2)                ▷ overlap: extend the last span
7      else
8          append y to c
9      end if
10     μ(i) ← index of the last span
11 end for
12 cap every span at 10 s of length                  ▷ scoring only, the file keeps its times
13 give each dropped cue the span of the cue before it
```

In split mode, and in the recut and joined paths of Section 10, the cues are not 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. Not merging at all is no good either.[2] So every mode merges with the same rule: a cue that starts inside the span before it is glued onto it, which is the condition in line 4 of Algorithm 1. On sorted cues that is the ordinary merge of overlapping cues. A cue that starts before the previous one is still left on its own, so a subtitle that starts over is not welded to the part before it.

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

$$t' = \operatorname{round}\big(t\,(1+\varepsilon) + \sigma_{\mu(i)}\big) \tag{11}$$

for both its start and its end. A cue whose new start s' is negative then has both times moved by $-s'$, so it keeps its length and starts at zero. 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, file permissions 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. A file that would not move is not written at all: when every offset is below `--min-shift`, 1 ms by default, and there is no stretch, the original is left as it was and the JSON line says `unchanged`.

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 canonical path, it's size, its modification time, the audio track number and the detector that heard it. The path is canonical so that a film reached as `./film.mkv` and again by its full path is listened to once, and so that the Docker container and a shell on the same share find the same profile.[3]

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, and the stream is checked after it is closed. That check is there because of a full disk.[4] A cache with fewer than 20 spans, or one that covers less than the whole film, is ignored and the film is listened to again.

[2] Before 2.3.1 these paths did exactly that, and it had its own cost. The split search does not let a cue start before the one ahead of it ends, so every pair of lines on screen at the same time forced a new part, and near the end of a file this left parts of one or two cues.

[3] Earlier versions hashed the path as it was given, so such a film was listened to twice and the container and the shell each kept a profile of their own.

[4] Without it, a disk that filled up during the write left a short file that was renamed into place anyway. On a two hour film 177 of 1 022 spans made it into the file, and a subtitle that had synced at $\Lambda = 55$ came back at 18 from then on.

4.8 Changes between 2.2.3 and 2.5.0

The changes that reach the numbers in this report are, by stage:

- *Detection.* Pieces of 180 s, down from 600 s, the three mixes probed in parallel, a shorter resampling filter, a channel map for single channel mixes, the 150 ms lead-in on Silero spans (Section 5.6), and the optional Whisper filter.
- *Search.* A second FFT plan reaching ±90 minutes, medians found by selection, a floor under the baseline spread of the lock statistic (Section 7.4), and ties between candidates broken by peak height.
- *Drift.* A continuous search over ratios between 0.94 and 1.06 (Section 8.5), and the choice between neighbouring framerates made after the split (Section 8.6). The chunked regression of 2.2.3 is gone.
- *Split.* An irregular grid with islands around far offsets that two slices found, or one found clearly; a jump to any offset at twice the penalty; half the penalty at chapter marks; a check that removes pieces nothing backs; boundaries that cannot be settled past their neighbours.
- *Decision and verdict.* A search over sixteen slices to ±90 minutes on every file; the split tried for unsure answers too; every answer in pieces judged on the finished file; a second, wider split when a piece is pressed against the edge of the fine window; the *nothing* threshold raised from 3.5 to 6.

Outside these stages, 2.5.0 adds `--from`, `--shift` with `--ratio`, `--threads`, `--min-shift`, chapters as split hints, and readers and writers for EBU STL, text STL, Scenarist SCC and iTunes Timed Text. The benchmark of Section 12 ran on the version described here.

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 180 s and decoded on one thread per core, up to eight, or on n threads with `--threads` n, each piece seeking to 3 s before its start and discarding the samples before it. With T the running time in seconds there are $\lceil T/180 \rceil$ pieces, forty for a two hour film. Short pieces keep every core busy on a short episode.

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, 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. The probes run at the same time, each through the detector, and the mix with the highest

$$Q = \underbrace{\frac{1}{F}\sum_{f=1}^{F} \max(p_f,\, 1-p_f)}_{\text{decisiveness}} \cdot \pi(\bar{p}), \qquad \pi(\bar{p}) = \begin{cases} \bar{p}/0.15 & \bar{p} < 0.15, \\ 1 & 0.15 \le \bar{p} \le 0.65, \\ (1-\bar{p})/0.35 & \bar{p} > 0.65, \end{cases} \tag{12}$$

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

$$g = \operatorname{clip}\Big(\frac{0.1}{L_{90}},\ 1,\ 12\Big), \tag{13}$$

where L_{90} is the 90th percentile of the RMS level over 10 ms frames, and clipped to $[-1, 1]$. A film that is quiet most of the time would pull the mean towards zero and be boosted into clipping in its loud scenes. The 90th percentile tracks how loud the film gets when something happens. It is taken from the frame levels the engine computes anyway, with a selection in linear time, so there is no second pass over the samples.

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 180 s has 5 625 windows, so each lane covers 704 windows, about 22.5 s, and a piece needs 704 calls instead of 5 625. Each lane starts with a zero state, so the model warms up eight times per piece 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** $W < 8 \cdot 64$ **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

The short pieces have a cost that has not been mesured. A two hour film has $40 \times 8 = 320$ cold lane starts, where ten minute pieces would give 96, and a line spoken in the first windows of a lane is heard late or not at all (Section 14.4).

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:

$$L_f = 0.995\, L_{f-1} + 0.005\, \ell_f, \qquad g_f = \mathrm{clip}(0.05/L_f,\ 0.5,\ 40), \tag{14}$$

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 $10\,\mathrm{ms}/0.005 = 2\,\mathrm{s}$.

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

$$p_f \leftarrow p_f \cdot \min\bigl(1,\ \mathrm{CV}_f/0.35\bigr). \tag{15}$$

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 left unfiltered, 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 ← 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 $\operatorname{len} r^k < 200$ 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 rest on the guess that nobody says anything in a fifth of a second and nobody breathes in less.

5.6 Lead-in

Subtitles are timed to come up a few frames before the first word, and Silero only calls a frame speech once the word is loud enough. An engine that lines the two up exactly therefore puts every line a little late. To make up for it, every span heard by Silero starts earlier by $\ell = 150$ ms, and never before the end of the span in front of it:

$$r_1^k \ \leftarrow\ \max\bigl(r_2^{k-1},\ r_1^k - \ell\bigr), \qquad r_2^0 := 0. \tag{16}$$

The step keeps the sequence valid, since each start stays at or after the previous end and before its own end. It is applied where the spans enter the alignment, after the cache, so the cache still holds what the detector heard and changing ℓ needs no new decoding. A reference from a subtitle or an embedded track is left alone, and so are spans from libfvad, which fires on the first frame with any voice in it.

The lead-in was first applied to both detectors, and every libfvad result then came out 150 ms early. The build without ONNX Runtime used for the sanitizer tests failed its whole accuracy suite by exactly that amount, which is how it was found.

How much of ℓ reaches the answer depends on the part of the search that decides it.

Proposition 5.1. *Suppose no start is clipped by* (16). *Then the bunching B of Section 7.3 computed on the shifted reference is that of the original reference translated by $-\ell$, and so is the onset channel of Section 6.4 up to the rounding of each start to its 40 ms bin. The block channel of a span of length L gains ℓ at its front, and its centre of mass moves by $-\ell/2$.*

Proof. Both the onset channel and B depend on the reference only through the set of span starts, and every start moves by $-\ell$. The block channel of a span is its indicator, which after the step covers $[r_1 - \ell, r_2)$; the centre of mass moves from $(r_1 + r_2)/2$ to $(r_1 - \ell + r_2)/2$. □

So the answer moves earlier by an amount between $\ell/2$ and ℓ, depending on how much the onsets dominate. In development, on 15 television episodes that carry a subtitle track timed to the episode by whoever made it, the mean distance between the engine's result and that track went from 192 to 109 ms, an improvement of 83 ms, inside the predicted range of 75 to 150. I did not expect a single constant to show up this clearly. alass and ffsubsync, run with their own audio detectors on the same episodes, were 114 and 139 ms away. These are development numbers on material that is not published, and the 150 ms is a single constant fitted to them. A detector that is late by an amount that depends on the speaker or on the loudness of the line would need something more careful than a constant.

5.7 Words as a filter

Silero is good at telling speech from music and less good at telling speech from singing, shouting crowds and dialogue in a language that is mumbled under effects. With `--whisper` and a model file, the engine runs `whisper-cli` from whisper.cpp [10] over the soundtrack and asks for token times aligned by dynamic time warping, read from the `t_dtw` field of its JSON output in units of 10 ms. Every token with a letter or a digit in it counts as a word at that time. A speech span r^k is kept when some word time w satisfies

$$r_1^k - 300\,\text{ms} \;\le\; w \;\le\; r_2^k + 300\,\text{ms}, \tag{17}$$

and dropped otherwise. If Whisper hears fewer than 50 words in the whole film, the filter is switched off and every span is kept. The words are cached next to the speech profile, under the name of the model, because a run of a larger model over a feature film takes minutes.

The filter only removes spans. It never moves one and never compares a word with the text of the subtitle, so the engine stays language agnostic in the sense of Section 2: Whisper decides where words are, and the subtitle can be in any language. The filter is not part of the benchmark, and there is no measurement in this report of what it does for accuracy.

5.8 How the detector got here

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, which gave 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 10% 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 of ten minutes made Proposition 4.1 true. They turned out to be too long for television: a 55 minute episode was six pieces, so a machine with more cores sat partly idle and one with fewer waited for a slow last piece. With three minute pieces and the mix probes run side by side, a cold run on such an episode with AC-3 5.1 went from 9.1 to 7.4 s.

A development build before 2.0.0 listened only to 30 windows spread over the film. Caches from that build record that they cover only part of the film; until 2.3.1 such a cache was loaded as if it covered all of it, and kept forever, since loading a cache refreshes its date.

Whether three minutes is the right length I honestly do not know. It was picked for the speed on episodes, and the price in cold lane starts is still unmeasured.

6 Searching for one offset

6.1 The exact sweep

Lemma 3.5 makes S in (8) 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 $\mathscr{B}$ of the $4KN$ trapezoid breakpoints. If $S \not\equiv 0$, its maximum over $\mathbb{R}$ is attained at a point of $\mathscr{B}$, and in particular at an integer.*

Proof. Continuity and piecewise linearity are inherited from each ϕ_{kn}. Between two consecutive points of $\mathscr{B}$ the function is linear, so its maximum over the closed segment is at an end. Outside $[\min \mathscr{B}, \max \mathscr{B}]$ 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. 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. 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

$E \leftarrow \emptyset$
for every pair (k, n) with $r_1^k - c_2^n \le D$ and $r_2^k - c_1^n \ge -D$ **do**
  $s \leftarrow h_k / \max(\operatorname{len} r^k, \operatorname{len} c^n)$; add $(\beta_1, +s), (\beta_2, -s), (\beta_3, -s), (\beta_4, +s)$ to E ▷ Lemma 3.5
end for
sort E by position
$v \leftarrow 0$; slope $\leftarrow 0$; last $\leftarrow 0$
for $(\beta, \Delta) \in E$ in order **do**
  $v \leftarrow v + \text{slope} \cdot (\beta - \text{last})$ ▷ $v = S(\beta)$
  if $|\beta| \le D$ **then** update the best (v, β) and the maximum of bucket $\lfloor (\beta + D)/1000 \rfloor$
  end if
  slope $\leftarrow$ slope $+\Delta$; last $\leftarrow \beta$
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 a range of $D = 15$ 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 hundreds of calls that drift and slice analysis need. LAPSE therefore runs the common path on FFTs, the ±90 minute searches for far parts included. The sweep is left with two jobs: references with fewer than four spans, and the per cue score curves of split mode, where one cue against the spans it can reach is cheap.

6.2 Why the maximum of S is not enough

The sweep finds the maximum of S exactly. The trouble is that S measures something slightly different from whether the offset is right. 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,

$$x = \bar{x} + \tilde{x}, \qquad \bar{x} = b_W * x, \tag{18}$$

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:

$$x \star y = \underbrace{\bar{x} \star \bar{y}}_{\text{hill}} + \bar{x} \star \tilde{y} + \tilde{x} \star \bar{y} + \underbrace{\tilde{x} \star \tilde{y}}_{\text{spike}}. \tag{19}$$

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. I did not expect this to matter as much as it did. The FFT search removes the hill before it looks for the peak.

The decomposition above is a heuristic, and the next subsection makes it exact under a model.

6.3 A point process view

Treat speech onsets as a point process on the time axis of the film. Dialogue comes in scenes, so the rate of onsets is itself random and varies on the scale of minutes; the natural model is a doubly stochastic, or Cox, process [5]. The reference and the subtitle are two noisy views of it.

Definition 6.3 (Onset model)**.** Let $\Lambda(t) \geq 0$ be a stationary random intensity with mean $\bar{\lambda}$ and covariance $C_\Lambda(u) = \operatorname{Cov}\big(\Lambda(t), \Lambda(t+u)\big)$, integrable with spectral density $P_\Lambda(f) = \int C_\Lambda(u) e^{-2\pi i f u}\, du$. Given Λ, the true onsets $\mathcal{N}$ form a Poisson process with intensity Λ. The detector reports each true onset independently with probability q_r and adds false alarms from an independent Poisson process of rate μ_r; this is the reference N_r. The subtitle contains each true onset at x independently with probability q_s, as a cue start at $x - \sigma^\star + \varepsilon$ with ε drawn independently from a density κ, and adds cue starts with no speech behind them from an independent Poisson process of rate μ_s; this is N_s.

Proposition 6.4 (Cross spectrum)**.** *Under Definition 6.3, the cross covariance density $c(u)$ of N_r at time $t+u$ and N_s at time t is*

$$c(u) = q_r q_s \Big[\underbrace{\bar{\lambda}\, \tilde{\kappa}(u - \sigma^\star)}_{\text{same onset}} + \underbrace{(C_\Lambda * \tilde{\kappa})(u - \sigma^\star)}_{\text{different onsets}} \Big], \qquad \tilde{\kappa}(v) = \kappa(-v), \tag{20}$$

and its Fourier transform is

$$\Gamma(f) = q_r q_s\, e^{-2\pi i f \sigma^\star}\, \overline{\hat{\kappa}(f)}\, \big(\bar{\lambda} + P_\Lambda(f)\big). \tag{21}$$

Proof. Write $E\,[dN_r(t+u)\, dN_s(t)]$ as a sum over pairs of a reported onset and a subtitled one. Pairs in which either point is a false alarm or a cue without speech contribute the product of the means, because those processes are independent of everything else, and they cancel when the product of the means is subtracted. A pair formed from one true onset x that is both reported and subtitled requires $x = t + u$ and $x - \sigma^\star + \varepsilon = t$, so $\varepsilon = \sigma^\star - u$; it occurs at rate $q_r q_s \bar{\lambda}$ with density $\kappa(\sigma^\star - u) = \tilde{\kappa}(u - \sigma^\star)$. A pair formed from two distinct true onsets y and x has, conditionally on Λ, product density $q_r q_s \Lambda(y)\Lambda(x)$, with $y = t + u$ and $x = t + \sigma^\star - \varepsilon$. Taking expectations gives $q_r q_s\big(\bar{\lambda}^2 + C_\Lambda(u - \sigma^\star + \varepsilon)\big)$ averaged over ε. The constant $q_r q_s \bar{\lambda}^2$ cancels against the same part of the product of the means, and the average of the rest over ε is $(C_\Lambda * \tilde{\kappa})(u - \sigma^\star)$. The transform follows from the shift and convolution theorems, with $\hat{\tilde{\kappa}} = \overline{\hat{\kappa}}$ for a real density. □

The offset sits entirely in the phase factor $e^{-2\pi i f\sigma^\star}$, and both terms carry it. They differ on where their energy lies. The first term is flat in f apart from the jitter $\hat\kappa$; it is the spike, a copy of κ at $\sigma^\star$, and its height grows with the number of onsets that both sides share. The second term has the spectrum P_Λ of the dialogue density, concentrated below the inverse correlation time of Λ, and it is the hill.

In the ensemble the hill also peaks at $\sigma^\star$. In one film it is a single sample path, and the false alarms are not Poisson. Score that the detector mistakes for speech comes in stretches with its own slowly varying rate, and that rate has no counterpart in the subtitle. Its power adds to the low frequencies of the reference alone, and the realised hill can have its top anywhere within its own width, which is minutes. The spike lives at frequencies where none of this happens.

The trend removal of Section 6.5 multiplies the cross spectrum by $|\hat g_W(f)|^2$ with $g_W = \delta - b_W$ (Proposition 6.9). The next proposition says how much of each term survives in the simplest case where it can be computed in closed form.

Proposition 6.5 (What survives of the hill). *Let $C_\Lambda(u) = v\,e^{-|u|/\tau}$, so that $P_\Lambda(f) = 2v\tau/\big(1+(2\pi f\tau)^2\big)$, and let b_W be the centred box of width W on the real line. The fraction of the energy of the hill term C_Λ that remains after the trend removal satisfies*

$$\frac{\int |\hat g_W(f)|^4 P_\Lambda(f)^2\,df}{\int P_\Lambda(f)^2\,df} \longrightarrow \frac{47}{10080}\Big(\frac{W}{\tau}\Big)^3 \qquad \text{as } W/\tau \to 0, \tag{22}$$

in the sense that the ratio of the two sides tends to 1.

Proof. By Parseval the denominator is $\int C_\Lambda^2 = v^2\tau$. With $\hat b_W(f) = \sin(\pi fW)/(\pi fW)$ and the substitution $x = \pi fW$, the numerator is

$$\frac{8v^2\tau^2}{\pi W}\int_0^\infty \Big(1-\frac{\sin x}{x}\Big)^4 \frac{dx}{\big(1+(2x\tau/W)^2\big)^2}.$$

Multiply the integrand by $(2\tau/W)^4$. The product increases to $(1-\sin x/x)^4 x^{-4}$ as $\tau/W \to \infty$, and that function is integrable, since it behaves like $x^4/6^4$ near zero and like x^{-4} at infinity. By monotone convergence the numerator is asymptotic to $\frac{8v^2\tau^2}{\pi W}\big(\frac{W}{2\tau}\big)^4 I$ with $I = \int_0^\infty (1-\sin x/x)^4 x^{-4}\,dx$. To evaluate I, take $W = 1$ and let $\psi = (\delta - b_1) * (\delta - b_1) = \delta - 2b_1 + b_1 * b_1$, whose transform is $(1-\sin \pi f/\pi f)^2$. Since ψ is even with $\int\psi = 0$, its second antiderivative Ψ is a continuous piecewise cubic with knots at $0, \pm\frac12, \pm1$ that vanishes outside $[-1,1]$. Its transform is $-\hat\psi/(4\pi^2 f^2)$, so Parseval and the substitution $x = \pi f$ give $\int\Psi^2 = \int \hat\psi^2/(16\pi^4 f^4)\,df = I/(8\pi)$. Integrating its square piece by piece gives $\int\Psi^2 = 47/40320$, hence $I = 47\pi/5040 \approx 0.0292966$. Dividing the numerator by $v^2\tau$ gives $\frac{I}{2\pi}(W/\tau)^3 = \frac{47}{10080}(W/\tau)^3$. □

Because of the cube, halving the width of the running mean keeps an eighth as much of the hill, and the spike, whose energy lies above $1/W$, barely notices. For the jitter densities that fit the two channels, a Gaussian of 170 ms for onsets (Proposition 6.8) and a triangle of half width 2 s for blocks, a numerical integration of $\int |\hat g_W|^4|\hat\kappa|^2 / \int|\hat\kappa|^2$ gives 0.964 for the onset channel with $W = 8$ s and 0.948 for the block channel with $W = 30$ s. Table 2 puts the two side by side for a few correlation times. The exponential covariance is a choice made for the closed form, and the numbers should be read as orders of magnitude.

What the filter does to a real curve is shown in Figure 3, for the film of Section 11 shifted by 33 s. The exact score S has its peak at the right place and a robust z of 7.2 (the statistic of (30)), and the strongest rival more than 5 s away reaches half the height of the peak above the median. The same two channel FFT curve computed without the trend removal has $z = 11.3$ and a rival at 31%. With the trend removal it has $z = 42.7$ and a rival at 9%. One film is an illustration, and the second step of that comparison is the ablation that Section 12.14 asks for over the whole benchmark.

	Hill energy kept, correlation time τ		
Channel, filter width	60 s	120 s	300 s
blocks, $W = 30$ s (spike kept 0.948)	5.7×10^{-4}	7.3×10^{-5}	4.7×10^{-6}
onsets, $W = 8$ s (spike kept 0.964)	1.1×10^{-5}	1.4×10^{-6}	8.8×10^{-8}

Table 2. The left side of (22), integrated numerically. For $\tau = 120$ s and $W = 30$ s the limit $\frac{47}{10080}(W/\tau)^3$ gives 7.29×10^{-5}.

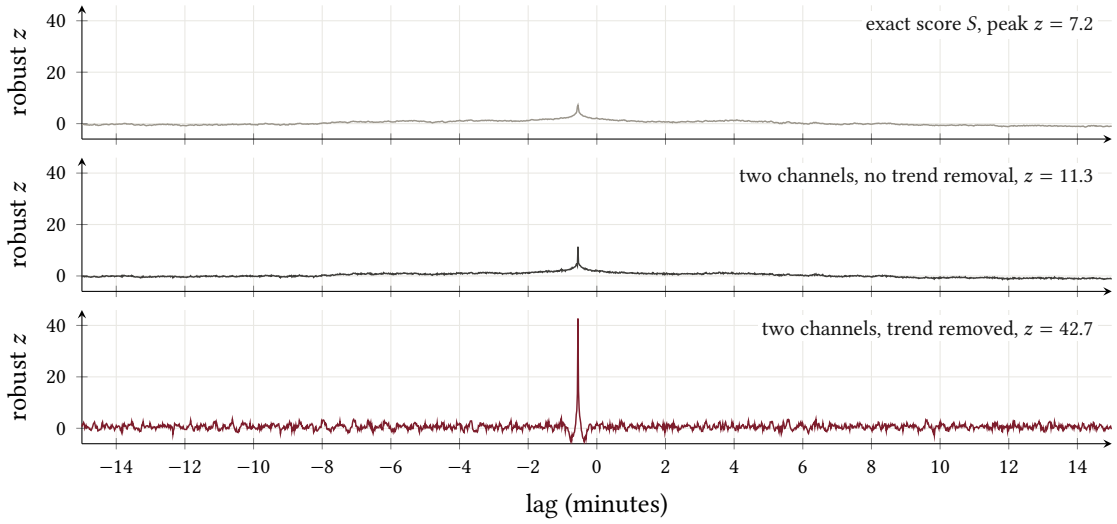


Figure 3. Three score curves over ±15 minutes for *(500) Days of Summer* with every cue 33 s late, all on the same robust scale, the height above the median in units of 1.4826 MAD. Each curve is pooled to its maximum per second. The top curve is the exact overlap score, the lower two are pass 1 of Algorithm 5.

6.4 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 6.6 (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

$$u_i^{\mathrm{b}} = \sum_k h_k \, \mathbf{1}[i \in B(r^k)], \qquad v_i^{\mathrm{b}} = \sum_n \frac{\mathbf{1}[i \in B(c^n)]}{m(c^n)}. \tag{23}$$

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:

$$u^{\mathrm{o}} = \kappa * \sum_k h_k \, \delta_{\lfloor r_1^k / G \rfloor}, \qquad v^{\mathrm{o}} = \kappa * \sum_n \delta_{\lfloor c_1^n / G \rfloor}. \tag{24}$$

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

$$(u^{\mathrm{b}} \star v^{\mathrm{b}})_j = \sum_n \sum_k h_k \, \frac{\mathrm{ov}(r^k, \, c^n + jG)}{\mathrm{len}\, c^n}, \tag{25}$$

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

Proof. $\sum_i v_i^{\mathrm{b}} u_{i+j}^{\mathrm{b}} = \sum_n \frac{1}{m(c^n)} \sum_k h_k \, |\, (B(c^n) + j) \cap B(r^k) \,|$. 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. □

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 6.8 (What the onset channel measures)**.** *Treating the kernel as continuous,* $(u^{\mathrm{o}} \star v^{\mathrm{o}})_j = \sum_n \sum_k h_k \, \gamma(r_1^k - c_1^n - jG)$*, 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.

6.5 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 6.9 (Filtering both signals filters the score)**.** *For finite sequences* x, y *and any filter* g *with finite support,*

$$(g * x) \star (g * y) = (g \star g) * (x \star y). \tag{26}$$

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

Proof. Let $\hat{x}(f) = \sum_i x_i e^{-2\pi i f i}$. The correlation $(x \star y)_j = \sum_i y_i x_{i+j}$, the convention of Section 6.4, has transform $\hat{x}(f)\,\overline{\hat{y}(f)}$. Filtering multiplies each transform by $\hat{g}$, so the correlation of the filtered sequences has transform $\hat{g}\hat{x}\,\overline{\hat{g}\hat{y}} = |\hat{g}|^2\,\hat{x}\overline{\hat{y}}$, and $|\hat{g}|^2$ is the transform of $g \star g$. □

In the language of Knapp and Carter [18], this is a generalised cross correlation with the fixed weighting $|\hat{g}|^2$. 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 $f = 1/W$ [12]. Hence

$$|\hat{g}(f)|^2 = \left(1 - \frac{\sin(\pi f W)}{W \sin(\pi f)}\right)^2, \tag{27}$$

which is 0 at $f = 0$, grows like $\left(\pi^2 (W^2 - 1) f^2 / 6\right)^2$ for small f, and is close to 1 for $f \gtrsim 1/W$. The engine's boxes have an even number of bins, 750 and 200, and cover $[i - W/2,\ i + W/2)$, so they sit half a bin early. Their transform carries the extra phase $e^{\pi i f}$, which turns (27) into $1 - 2\cos(\pi f)\,\hat{b}_W + \hat{b}_W^2$ and adds about $\pi^2 f^2$ to it; that term dominates only below $f = 6/(\pi W^2)$, a period of hours for the block channel. Filtering both channels therefore high pass filters the score curve over lags, with a cutoff of about one cycle per W. 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.

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

$$\rho_j^{\mathrm{b}} = \frac{(u^{\mathrm{b}} \star v^{\mathrm{b}})_j}{\|u^{\mathrm{b}}\|\,\|v^{\mathrm{b}}\|}, \qquad \rho_j^{\mathrm{o}} = \frac{(u^{\mathrm{o}} \star v^{\mathrm{o}})_j}{\|u^{\mathrm{o}}\|\,\|v^{\mathrm{o}}\|}, \tag{28}$$

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 6.8.

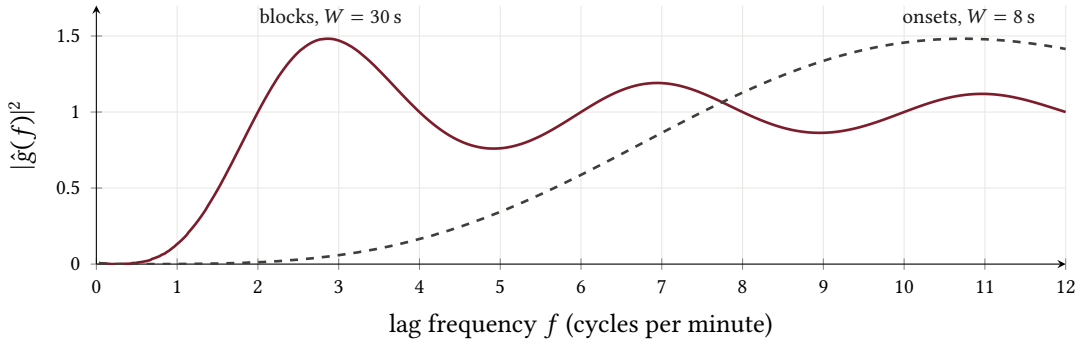


Figure 4. Response of the trend removal on the score curve, (27), with lags in 40 ms bins (1 500 per minute). Anything in the score that changes more slowly than once per 30 s (blocks) or per 8 s (onsets) is removed. Values above 1 come from the side lobes of the box average.

6.6 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 6.10. *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 6.11. *Let x be supported on $[0, L_x)$, y on $[0, L_y)$, and $N_{\mathrm{fft}} \geq \max(L_x, L_y) + J$. For $|j| \leq J$ 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 $i + j + N \geq N - J \geq L_y$ and $i + j - N < L_x + J - N \leq 0$, neither can. □

For a reach of J bins the engine takes

$$N_{\mathrm{fft}}(J) = 2^{\lceil \log_2(1.25\, n_r + J + 64) \rceil} \tag{29}$$

and truncates the subtitle channels to $N_{\mathrm{fft}} - J - 8$ bins, which satisfies the proposition on both sides. The factor 1.25 leaves room for a subtitle that runs longer than the speech in the film, as happens with a file timed at 23.976 against a 25 fps video. There are two reaches. The close plan has $J = 15\,\mathrm{min}/G = 22\,500$ bins. The far plan has $J = 90\,\mathrm{min}/G = 135\,000$ bins and is built the first time a search asks for more than 15 minutes. For the film of Section 11, $n_r = 139\,130$, so the close plan has $N_{\mathrm{fft}} = 262\,144$ and the far plan 524 288.

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:

$$z = \frac{x_{\text{peak}} - \operatorname{med}(x)}{1.4826\ \operatorname{MAD}(x)}, \tag{30}$$

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 ignores its own shoulders [25]. Both statistics are found by selection in linear time (over the far reach that is close to 100 000 numbers, four passes per query, so it adds up). 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 $\hat{u}^{\mathrm{b}}, \hat{u}^{\mathrm{o}}$ and norms for the plan that covers the range; cues c
Output: candidate offsets with scores z

build $v^{\mathrm{b}}, v^{\mathrm{o}}$ from c; subtract running means; compute norms and $\hat{v}^{\mathrm{b}}, \hat{v}^{\mathrm{o}}$
for each pass $(\alpha_{\mathrm{b}}, \alpha_{\mathrm{o}}, s)$ in the table of Section 6.6 **do**
  $X_f \leftarrow \left(\alpha_{\mathrm{b}}\, \hat{u}^{\mathrm{b}}_f \overline{\hat{v}^{\mathrm{b}}_f} / \|u^{\mathrm{b}}\|\|v^{\mathrm{b}}\| + 1.15\, \alpha_{\mathrm{o}}\, \hat{u}^{\mathrm{o}}_f \overline{\hat{v}^{\mathrm{o}}_f} / \|u^{\mathrm{o}}\|\|v^{\mathrm{o}}\|\right) e^{-2\pi^2 s^2 (f/N_{\mathrm{fft}})^2}$
  $x \leftarrow$ inverse transform of X, restricted to lags $|j| \le J$
  take the 10 highest peaks, ±2.5 s apart; score each by (30)
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 $O(N_{\mathrm{fft}})$ work per peak, with the radix 2 lengths that the Cooley and Tukey algorithm [4] handles best. 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. Until 2.3.0 every search beyond 15 minutes fell back to the sweep. Against speech the plain overlap picks up next to nothing that far out, and a file with half its cues 20 minutes away came out shifted, solid, and half wrong. That case is why the far plan exists.
> The median and MAD of (30) were found by sorting until 2.5.0. Selection gives the same numbers, and on eight test films it cut the CPU time of a cached run from 5.6 to 4.3 s.

7 Refinement and the lock statistic

The FFT search is fast and resolves offsets to 40 ms. It also scores something other than 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,*

$$|S'(\sigma)| \le \Lambda_S := 2 \sum_{n=1}^{N} \frac{1}{\operatorname{len} c^n}. \tag{31}$$

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. 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 What the refinement estimates

Proposition 3.6 turns the refinement into a familiar estimator. Call a pair (k, n) *active* at σ if $\operatorname{dist}(\sigma, P_{kn}) < \ell_{kn}$, so that its trapezoid is non-zero there, and write $w_{kn} = h_k / L_{kn}$.

Proposition 7.3 (Interval median)**.** *On any interval J of offsets on which the set A of active pairs does not change,*

$$S(\sigma) = \sum_{(k,n) \in A} w_{kn} \ell_{kn} \; - \sum_{(k,n) \in A} w_{kn} \operatorname{dist}(\sigma, P_{kn}), \qquad \sigma \in J. \tag{32}$$

The maximisers of S on J are therefore the minimisers of a weighted sum of distances to intervals. When every plateau is a single point s_{kn}, as it is for a cue and a speech span of equal length, they are the weighted medians of the points s_{kn} with weights w_{kn}, provided a weighted median lies in J; otherwise the maximiser is the end of J nearest to them.

Proof. On J every active pair has dist $< \ell$, so the positive part in (7) can be dropped, and every inactive pair contributes zero. Summing (7) over A gives the display. For point plateaus, $\sum w \, |\sigma - s|$ is convex in σ and minimised over $\mathbb{R}$ exactly by the weighted medians, so its minimisers over the interval J are those medians that lie in J, or the nearest end of J when none does. □

The exact score, then, behaves like a median. Each well placed pair pulls the the answer towards its plateau with a force that does not grow with distance, and a pair further away than its own length stops pulling at all. In the language of Hampel [11] its influence is bounded and it redescends to zero. A cue on a line that is not in the audio, or a detector span that is really music, can move the answer by at most one rank in the median, however far away it sits. Like a median, it gives up efficiency under Gaussian noise.

Proposition 7.4 (Efficiency)**.** *Let M cues each be matched to one speech span of the same length, with plateau points $s_n = \sigma^\star + e_n$ and weights w_n. Let the pairs (w_n, e_n) be independent and identically distributed, with w_n bounded, positive and independent of e_n, and e_n symmetric with a density f that is continuous and*

positive at 0. Suppose the active set is constant on a fixed neighbourhood of $\sigma^\star$. Then the maximiser $\hat\sigma$ of S on that neighbourhood satisfies

$$\sqrt{M}\,(\hat\sigma - \sigma^\star) \xrightarrow{\;d\;} \mathcal{N}\Big(0,\ \frac{\mathbb{E}[w^2]}{4f(0)^2\,\mathbb{E}[w]^2}\Big). \tag{33}$$

For equal weights and Gaussian e_n with standard deviation s the variance is $\pi s^2/(2M)$, and the efficiency against the sample mean, which attains the Cramér and Rao bound s^2/M in that model, is $2/\pi \approx 0.64$.

Proof. By Proposition 7.3, $\hat\sigma$ is a weighted median of the s_n, the M-estimator with $\psi(e, w) = w\ \mathrm{sign}(e)$ [13]. Its expected score at a displacement t is $\mathbb{E}[w]\,(1 - 2F(t))$, with derivative $-2f(0)\mathbb{E}[w]$ at $t = 0$, and the variance of the score at the true value is $\mathbb{E}[w^2]$. The standard asymptotics of M-estimators with a monotone ψ give the variance $\mathbb{E}[w^2]/\big(2f(0)\mathbb{E}[w]\big)^2$. For a Gaussian, $f(0) = 1/(s\sqrt{2\pi})$. □

With $s = 150$ ms and $M = 1\,000$ matched cues, the standard deviation of the refined offset is $150\sqrt{\pi/2}/\sqrt{1000} \approx 5.9$ ms. Errors of tens of milliseconds, which the worked example and the benchmark both show, are therefore bias, from a detector that fires late and subtitlers who lead the speech, and that is the error the lead-in of Section 5.6 goes after.

The assumptions are strong, of course. Real cues and spans are rarely the same length, so most plateaus have width, and an interval median has a flat top wherever the plateaus overlap; what carries over to real films is the shape of the estimator, and the variance formula needs the assumptions.

7.3 Bunching

Speech starts carry most of the timing information (Proposition 6.8), 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

$$B(\sigma) = \frac{1}{N}\sum_{n=1}^{N} \exp\Big(-\frac{d_n(\sigma)^2}{2\tau_B^2}\Big), \qquad \tau_B = 450\ \text{ms}, \tag{34}$$

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, $O(\sigma) = S(\sigma)/N$, which also lies in $[0, 1]$, since a cue's pairs together cover at most the cue.

7.4 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

$$\mu_B = \frac{1}{96}\sum_i B(\xi_i), \quad s_B^2 = \frac{1}{96}\sum_i \big(B(\xi_i) - \mu_B\big)^2, \qquad z_B(\sigma) = \frac{B(\sigma) - \mu_B}{\max(s_B,\ 1/N)}, \tag{35}$$

and the same for O. The baseline depends only on the cue set, so it is computed once per set and cached. The cache key is the address, size, first cue and last cue of the set.[5]

Why the floor $1/N$? When the cues of a slice run past the end of the reference, which happens at the end of a subtitle that is longer than its film, every random drop scores zero, the spread is zero, and whatever the real offset scores divides by nothing. One cue moving onto or off a speech start changes B or O by at most $1/N$, so the floor says that the baseline cannot claim to be steadier than a single cue.

[5] The first cue was added to the key late. Without it, a freed slice and a new one of the same size, allocated at the same address and ending on the same cue, were found to share a baseline.

Proposition 7.5 (Invariance and range). *z_B does not depend on the weights h_k. z_O is unchanged when every h_k is multiplied by the same $\lambda > 0$, as long as the floor is not active, and both are unchanged when the subtitle and the reference are moved by the same amount of time. Both satisfy $|z| \le N$.*

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. For the bound, B and O lie in $[0, 1]$, so $|B - \mu_B| \le 1$, and the denominator is at least $1/N$. □

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.

7.5 Slices, and the statistic itself

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 the floor $1/N_j$ for a slice of N_j cues. With $z_{B,j}$ and $z_{O,j}$ the scores of slice j,

$$\zeta_j(\sigma) = \operatorname{clip}\Big(\tfrac{1}{2}\big(z_{B,j}(\sigma) + z_{O,j}(\sigma)\big),\ -1,\ 3\Big). \tag{36}$$

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.

The whole file scores and the slice scores are then added up with fixed weights.

Definition 7.6 (Lock statistic). With $z^+ = \max(0, z)$,

$$\Lambda(\sigma) = 0.6\,\big(z_B^+(\sigma) + z_O^+(\sigma)\big) + 1.6 \cdot \frac{1}{8}\sum_{j=1}^{8} \zeta_j(\sigma). \tag{37}$$

It is reported as `sigma`, clipped below at zero. For fewer than eight cues, or a reference of fewer than eight spans, it is zero.

Proposition 7.7. *$-1.6 \le \Lambda \le 1.2N + 4.8$, 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 $z_B^+ + z_O^+ \ge 5.3$; with the slice term at zero it needs* 13.3.

Proof. The first term lies in $[0, 1.2N]$ by Proposition 7.5. 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. I set the weights 0.6 and 1.6, the clip limits and τ_B by hand on the benchmark films (Section 14), which is a weakness and I know it.

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, which is then normal with variance 2, would have a probability of about 3×10^{-21}. They are neither independent nor normal: the random offsets sample one film

whose structure repeats, and the same few scenes influence both scores. The thresholds were chosen from outcomes, without a distribution behind them. Subtitles for an unrelated film score between 3 and 6, and that is why the *nothing* threshold sits at 6.

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 $D \le 90$ min
Output: offset $\hat{\sigma}$, lock $\Lambda(\hat{\sigma})$, confidence, margin

if the reference has at least four spans **then**
  $\mathscr{C} \leftarrow$ FftCandidates(c) with the plan that reaches D, $\cup\{0\}$ ▷ Algorithm 5; 0 lets a correct file win
  rank $\mathscr{C}$ by Λ, ties by peak z; 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 $(\Lambda(\sigma), z)$ 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. A slice of fewer than eight cues has $\Lambda = 0$ for every candidate, and the height of the FFT peak breaks the tie.

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,

$$\Lambda_{\text{sweep}} = \operatorname{clip}\Big(\frac{M - \operatorname{med}(\text{bucket maxima})}{1.4826\ \operatorname{MAD}(\text{bucket maxima})},\ 0,\ 99\Big). \tag{38}$$

This is the z of (30) applied to the upper envelope of S, and it reads on a different scale from Λ. The restart and joined modes of Section 9.8 search their parts with the far plan and so use Λ as well, which leaves the sweep statistic for references with fewer than four spans.

> **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, which ignores it; slice agreement covers the case it was meant for, a peak that fits only one part of the film. Λ replaced the ratio based measures in 2.0.0.
> The lock itself had teething problems. It had no floor on the spread at first, and a slice whose cues ran past the end of the reference came back with a lock of 31 million. The *nothing* threshold started at 3.5 and was raised to 6 once subtitles for an unrelated film were seen to score up to 6. And for a slice of fewer than eight cues, where every candidate has $\Lambda = 0$, the winner used to be whichever candidate the sort happened to leave first, which for more than sixteen candidates is not the order they came in. The restart and joined modes also searched their parts by the sweep for a while, so their verdicts were judged on the sweep scale against thresholds set for Λ.

8 Drift

8.1 The ratios that occur

Under the drift model (2), a subtitle timed for one framerate 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 3 lists the set P that LAPSE tries first.

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 8 s
30 and 25	1.200000	+20.0%	24 min

Table 3. 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; so do their inverses, 0.96 and 0.95904. Any method that estimates ρ has to separate values that close, and Section 8.6 shows a case where the obvious method cannot.

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 and have too few cues to lock. A short list of known ratios avoids the dilemma.

The chunked method was the only drift method for a while, and its story is a little embarrassing. Version 2.1, on 8 July, cut the profiles into chunks of 15 minutes from 7.5 minutes in, found a lag per chunk by FFT with a parabolic peak fit, and put a weighted least squares line through the lags. 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 framerates it was still about 67 s off; Proposition 8.1 above is why it could not have been rescued for the large ratios. The ratio search replaced it as the default in the same change. It stayed on as the fallback of `ols` mode until the continuous search of Section 8.5 took over, and it's code was removed.

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
```

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,

$$\hat{\varepsilon} = \operatorname{med}\left\{ \frac{\sigma_j - \sigma_i}{t_j - t_i} \;:\; i < j \right\}, \qquad \hat{\sigma}_0 = \operatorname{med}_j(\sigma_j - \hat{\varepsilon} t_j), \tag{39}$$

where the code takes, of the sorted slopes, the one at position $\lfloor m/2 \rfloor$ counting from zero, for m slopes. Its breakdown point is $1 - 1/\sqrt{2} \approx 29\%$ [29]: 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.

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

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 3. 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 $|\hat{\varepsilon}| > 10^{-5}$, and snaps $1 + \hat{\varepsilon}$ to the nearest ratio in P if one lies within 0.004.

8.5 Ratios between the pairs

A capture that ran a little fast, or a file that someone sped up by an odd amount, has a ratio that is not in P. For these the engine searches an interval $[\rho_-, \rho_+]$ in three stages, shown in Algorithm 8. The first two stages scan a grid and use the height z of the best FFT peak, which costs one query of Algorithm 5 per ratio. The third refines with a golden section search [16] on Λ itself.

Algorithm 8. Continuous ratio search

Input: cues c, $T = \max(60\,\mathrm{s},$ last cue end$)$, interval $[\rho_-, \rho_+]$
Output: ratio $\hat{\rho}$, offset, Λ

```
δ ← max(10^−4, 2400 ms/T)
ρ_1 ← arg max{z(ρc) : ρ = ρ_− + iδ ≤ ρ_+}        ▷ best FFT peak per ratio
ρ_2 ← arg max{z(ρc) : ρ = ρ_1 − δ + iδ/6 ≤ ρ_1 + δ}
[a, b] ← [ρ_2 − δ/6, ρ_2 + δ/6]; φ ← (√5 − 1)/2
x_1 ← b − φ(b − a);   x_2 ← a + φ(b − a)        ▷ Λ by Algorithm 6 on the scaled cues
for 12 iterations do
    if Λ(x_1) > Λ(x_2) then b ← x_2; x_2 ← x_1; x_1 ← b − φ(b − a)
    else a ← x_1; x_1 ← x_2; x_2 ← a + φ(b − a)
    end if
end for
```

11 **return** $\hat{\rho} = (a + b)/2$ with its offset and Λ

The grid step is chosen in units of the file. A ratio error of δ moves the last cue by δT against the first, so a step of $2400/T$ means that neighbouring grid ratios place the end of the file 2.4 s apart, and the nearest grid ratio is never more than 1.2 s out at the end. By Proposition 8.1 that keeps the true peak at a usable height on the grid, and the second scan at a sixth of the step narrows it to 200 ms.

Proposition 8.3 (Resolution of the golden section stage)**.** *If Λ is unimodal on the bracket, the returned ratio is within*

$$\frac{\delta}{6}\,\phi^{12} \;\approx\; 5.2 \times 10^{-4}\,\delta \tag{40}$$

of the maximiser in the bracket. For $\delta = 2400/T$ the end of the file is then placed within 1.24 ms *of where the maximiser would place it, whatever the length of the file.*

Proof. Each iteration keeps the subinterval that must contain the maximum of a unimodal function and shrinks the bracket by the factor ϕ, while one of the two interior points carries over, so each iteration costs one evaluation. After 12 iterations the bracket has width $(\delta/3)\phi^{12}$, and its midpoint is within half of that of any point in it. Multiplying $(\delta/6)\phi^{12}$ by T with $\delta = 2400/T$ gives $400\phi^{12}$ ms $\approx$ 1.24 ms. □

The hypothesis is the weak spot. Λ as a function of ρ has a clear peak and is not smooth at the scale of the bracket: in Figure 5 it moves by a unit between neighbouring points 5×10^{-5} apart, which is within the estimation error of the baseline (Section 7.4). The golden section stage finds a local maximum among these wiggles, which at this resolution is as good as any. The full search over $[0.94, 1.06]$ costs, for a film of 95 minutes, 285 FFT queries on the grid, 13 on the second scan and 15 calls to Algorithm 6, a few seconds in all. The narrow search of ± 0.003 around the slope of Section 8.4, which auto mode runs on every drifting file, costs a twentieth of the grid part.

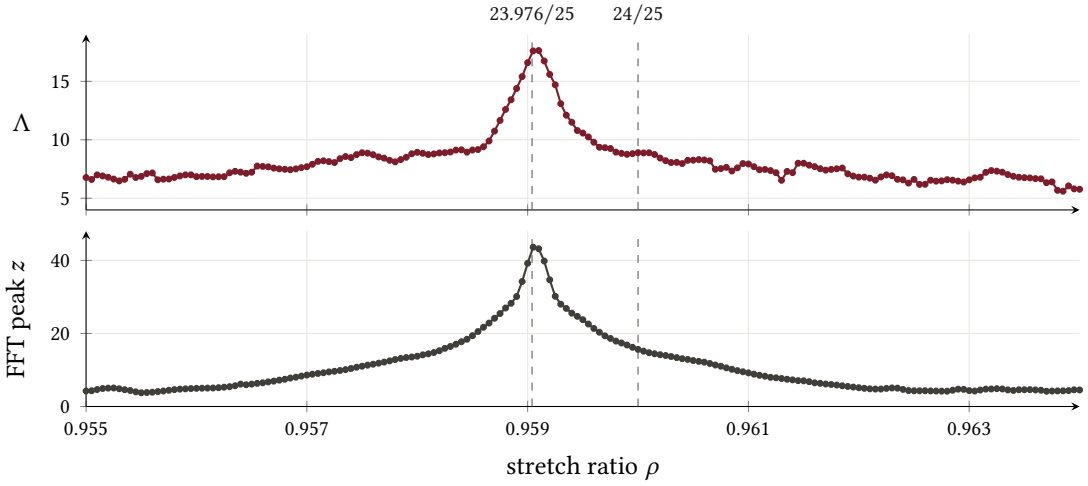


Figure 5. The lock Λ (top) and the height of the best FFT peak (bottom) of the drifting copy of Section 11, scaled by 181 ratios 5×10^{-5} apart. The copy was made with 25/23.976, so 23.976/25 undoes it. The neighbouring pair 24/25 lies on the flank of the peak.

8.6 Drift and a cut together

A PAL release of a director's cut both drifts and jumps, and the Theil and Sen line of Section 8.4 is then fitted to slices that do not lie on one line. The effect can be computed exactly.

Proposition 8.4 (Bias from a cut)**.** *Let eight slices have equally spaced midpoints $t_j = t_1 + (j-1)h$ and offsets $\sigma_j = \sigma_0 + \varepsilon t_j + \Delta\,\mathbf{1}[j \geq 5]$, a drift with a jump of $\Delta \neq 0$ between the fourth and fifth slice and no other error. Then the slope of* (39) *is*

$$\hat{\varepsilon} = \varepsilon + \frac{\Delta}{6h}. \tag{41}$$

Proof. Of the 28 pairwise slopes, the 12 within the first four or within the last four slices equal ε. The 16 that straddle the jump equal $\varepsilon + \Delta/(dh)$, where $d = j - i \in \{1, \dots, 7\}$ occurs $1, 2, 3, 4, 3, 2, 1$ times. Take $\Delta > 0$. In increasing order come the 12 values ε, then $\varepsilon + \Delta/7h$ once, then $\varepsilon + \Delta/6h$ twice, at positions 13 and 14 counting from zero, so position $\lfloor 28/2 \rfloor = 14$ holds $\varepsilon + \Delta/6h$. For $\Delta < 0$ the 16 straddling values come first, in increasing order $\Delta/h, \Delta/2h, \dots$ with the same multiplicities, so positions 0 to 12 hold $d \le 5$ and positions 13 and 14 hold $d = 6$; position 14 is again $\varepsilon + \Delta/6h$. The usual median, the mean of positions 13 and 14, gives the same value. □

Take a 45 minute episode, so $h \approx 337$ s, with a cut of $\Delta = 4$ s halfway. The bias is $4/(6 \cdot 337) \approx 2.0 \times 10^{-3}$, about twice the gap of 9.6×10^{-4} between $24/25$ and $23.976/25$. When the bias points towards the neighbour, snapping to the nearest ratio in P picks the wrong one, a ratio 0.1% off that puts the end of the episode 2.7 s out, and the split that follows moves the pieces by odd amounts to make up for it. I ran into exactly this in development: a subtitle stretched by $25/24$ with a 4 s cut halfway came out stretched by $23.976/25$ on most of the episodes it was tried on.

The slope cannot tell the two ratios apart because the file has not yet been cut, and both the slices and the plain ratio search measure it with one offset. So once auto mode has settled on a stretch $1 + \hat{\varepsilon}$, it tries every ratio in P within 0.004 of it. For each, it scales the cues, finds the offset, runs the fine split search of Section 9 as it would anyway, and computes Λ of the file with the pieces moved; the ratio with the highest value wins. The drifting copy of Section 11 shows the comparison on a file without a cut. With $23.976/25$ the split finds nothing to do and the file reaches $\Lambda = 17.5$. With $24/25$ the split cuts the file into three pieces at -0.99, -2.89 and -5.12 s, a staircase that follows the residual drift, and reaches 11.6. Without the comparison, a file snapped to the wrong ratio would be kept as a staircase.

On 20 development episodes and films with a drift and a cut, an earlier version without the comparison had 90% of lines within 250 ms on 8 of them, and the comparison brings that to 14 of 20.

8.7 The explicit mode

The mode `ols` skips the decision of Section 10. It runs Algorithm 7, then Algorithm 8 over ± 0.003 around the winner, or over $[0.94, 1.06]$ when no ratio in P reaches a lock of 6, and keeps the continuous result if it beats the best ratio in P by at least one unit of Λ. The name is a leftover from the regression it once ran, and I have not got round to changing it.

9 Split alignment

A recut film, a television capture with advert breaks, or a director's cut against a theatrical subtitle needs an offset that changes along the file. This section solves (9) on a grid, then cleans up the solution, and finally treats two structures that look like splits and have simpler answers: files whose timestamps start over, and videos that hold two parts.

9.1 The grid

Offsets are restricted to a finite, sorted set $\Sigma = \{\varsigma_0 < \varsigma_1 < \cdots < \varsigma_{M-1}\}$. Its core is a regular grid around an anchor σ_0, normally the global offset from Algorithm 6:

$$\Sigma_0 = \{\sigma_0 - W + m\delta \;:\; m = 0, \dots, 2W/\delta\}. \tag{42}$$

Two configurations are used. The recut search has $W = 600$ s and $\delta = 100$ ms, so $|\Sigma_0| = 12\,001$. The fine search inside the shifted and drifting paths has $W = 45$ s and $\delta = 25$ ms, so $|\Sigma_0| = 3\,601$.

A file with a long scene cut out of it needs an offset more than ten minutes from the anchor, and widening W costs memory for every cue. So the recut search keeps W and adds islands to the grid. The cues are cut into sixteen slices twice, and each slice is aligned on its own, once within ± 15 minutes and once within ± 90 minutes. A slice offset is *backed* when a second slice found the same offset within 1 s, or when the slice locked on by itself with $\Lambda \ge 7$. Every backed offset o from the ± 90 minute slices that lies outside Σ_0 adds the island $\{o - 45\text{ s} + m\delta\}$ of 901 points, and Σ is the sorted union.

The backing is there because a quiet slice near the end of a film finds a busy conversation an hour away about as clearly as its own place, and the split search, which scores how much dialogue the cues land on, will happily take such an offset. The escape for a lone slice with $\Lambda \ge 7$ is for a short far piece that only one slice covers (both cases are told in the history box at the end of this section).

With at most sixteen islands, $|\Sigma| \le 12\,001 + 16 \cdot 901 = 26\,417$.

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 $\sigma_{n-1} \le \sigma_n + \gamma_n$, 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, define for each m the largest index that the previous cue may take,

$$a_n(m) = \max\{m' \;:\; \varsigma_{m'} \le \varsigma_m + \gamma_n\}, \tag{43}$$

with $a_n(m) = -1$ when the set is empty. Since Σ is sorted, a_n is non-decreasing in m, and the constraint of Lemma 9.1 on the grid reads $m_{n-1} \le a_n(m_n)$. The constraint is exact on an irregular grid.[6]

9.3 The recursion

Three things can happen between one cue and the next. The offset can stay, at no cost. It can change to any offset the order constraint allows, at the penalty p_n, which is p in general and $p/2$ when a chapter mark of the video falls between the two cues (Section 9.5). Or it can change to any offset at all, at $2p$, whether or not the order is kept.

The last transition is for a scene with dialogue that has been cut from the video. The subtitle still has those lines, and the offset has to drop by the length of the scene in one go. The order constraint only lets it drop by the gap between two cues, so without the jump the search can only follow the cut as a staircase down through the nearest gaps. The lines of the missing scene still have nowhere right to go, but the rest of the file does. With 40 s cut ten minutes into a test film, the file comes out with 9 cues off, the ones just before the cut, where speech this dense scores nearly as well either way.

Definition 9.2. Let $f_n(m) = \sum_k \phi_{kn}(\varsigma_m)$ be the score of cue n at grid point m, and let $V_n(m)$ be the best value of

$$\sum_{i \le n} f_i(m_i) - \sum_{2 \le i \le n} \pi_i(m_{i-1}, m_i), \qquad \pi_i(m', m) = \begin{cases} 0 & m' = m,\ \gamma_i \ge 0, \\ p_i & m' \ne m,\ m' \le a_i(m), \\ 2p & \text{otherwise}, \end{cases}$$

over all $m_1, \dots, m_n$ with $m_n = m$.

The case $m' = m$ with $\gamma_i < 0$, two cues that already overlap in the file, falls under the third line, so staying at the same offset across an overlap costs $2p$.

Theorem 9.3. *$V_1 = f_1$, and for $n \ge 2$, with $V_{n-1}^{*} = \max_{m'} V_{n-1}(m')$,*

$$V_n(m) = f_n(m) + \max\Big(V_{n-1}(m) + \iota_n,\ \max_{m' \le a_n(m)} V_{n-1}(m') - p_n,\ V_{n-1}^{*} - 2p \Big), \tag{44}$$

where $\iota_n = 0$ if $\gamma_n \ge 0$ and $\iota_n = -\infty$ otherwise, so the first option is only available when $\gamma_n \ge 0$. The optimum of the grid problem is $\max_m V_N(m)$, and a maximising sequence is recovered by following the choices backwards. The recursion takes $O(N|\Sigma|)$ time beyond the per cue curves f_n, and $N|\Sigma|$ back pointers of 16 bits.

[6] The regular grid of 2.0.0 computed a_n by an integer division of the gap by the step.

Proof. The penalty π_i depends only on m_{i-1} and m_i, so the objective is a sum of terms that each couple two consecutive indices, and Bellman's principle applies [3]: in an optimal sequence ending at $m_n = m$, the prefix ending at m_{n-1} is optimal among sequences ending there, since replacing it by a better one changes no other term. It remains to show that the maximum over m' of $V_{n-1}(m') - \pi_n(m', m)$ equals the outer maximum in (44). The first option is the term $m' = m$ when it is free. Every m' that the second line of π_n admits appears in the second option at the right penalty. Since $a_n(m) \geq m$ exactly when $\gamma_n \geq 0$, the second option also contains $m' = m$ in that case, at the penalty p_n, and the first option dominates it; when $\gamma_n < 0$ the index m is in neither of the first two options. Every m' that falls under the third line appears in the third option at $2p$. The third option also contains indices that the first two already cover at a penalty of at most p, and for those it gives a lower value, so it cannot change the maximum. So the maximum is attained by the right penalty in every case. For the cost: since a_n is non-decreasing in m, the prefix maximum $\max_{m' \leq a_n(m)} V_{n-1}(m')$ can be carried along as m increases, adding at most one new index at a time, and V^*_{n-1} is one pass over the previous row. Each row therefore costs $O(|\Sigma|)$. One back pointer per cue and grid point recovers the sequence, and $|\Sigma| \leq 26\,417 < 2^{16}$. □

For a film with 1 500 cues, the recut grid without islands needs 1 500 × 12 001 back pointers of two bytes, about 36 MB. At a 1 ms step the same window would need 7.2 GB at the four bytes per pointer of the first implementation, which is what it tried to allocate. Even at a 100 ms step, four byte pointers gave a file of 5 000 cues 240 MB of them, hence the 16 bits. Algorithm 9 gives the full procedure.

Algorithm 9. 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

$\Sigma \leftarrow \Sigma_0$; **if** $W = 600$ s **then** add the islands of backed far slice offsets (Section 9.1)
$V \leftarrow f_1$
for $n = 2, \ldots, N$ **do**
  $\gamma \leftarrow c^n_1 - c^{n-1}_2$; $p_n \leftarrow p/2$ if a chapter mark lies between the cues, else p
  best $\leftarrow -\infty$; $a \leftarrow -1$; $V^* \leftarrow \max V$ with its index
  for $m = 0, \ldots, |\Sigma| - 1$ **do**
    while $a + 1 < |\Sigma|$ and $\varsigma_{a+1} \leq \varsigma_m + \gamma$ **do** $a \leftarrow a + 1$; update best, best_{arg} with $V[a]$
    end while
    if $a < 0$ **then** $V'[m] \leftarrow -\infty$
    else if $\gamma \geq 0$ and $V[m] \geq \text{best} - p_n$ **then** $V'[m] \leftarrow f_n(m) + V[m]$; $\text{back}[n][m] \leftarrow m$ ▷ stay
    else $V'[m] \leftarrow f_n(m) + \text{best} - p_n$; $\text{back}[n][m] \leftarrow \text{best}_{\text{arg}}$ ▷ split in order
    end if
    if $V^* - 2p > V'[m] - f_n(m)$ **then** $V'[m] \leftarrow f_n(m) + V^* - 2p$; $\text{back}[n][m] \leftarrow \arg\max V$
    end if ▷ jump
  end for
  $V \leftarrow V'$
end for
follow back pointers from $\arg\max V$; convert grid indices to offsets
CheckRuns; SettleBoundaries ▷ Section 9.4

One detail of Algorithm 9 departs from (44). A grid point with no admissible predecessor, $a < 0$, is set to $-\infty$ before the jump is considered, and the jump can then lift it again. In (44) the same point takes the jump directly. The two agree.

9.4 Checking runs and settling boundaries

Runs. The grid solution is only good to one grid step. Each maximal run of cues with the same offset is shifted by that offset and passed once more to the single offset search. In the recut search the run must also be backed by something. It is kept if it locks on by itself with $\Lambda \geq 7$, if a neighbouring run lies within 45 s of it, or if the anchor, a backed slice offset or another run lies within 1 s of it. A run that passes none of these is given the offset of whichever neighbour it scores better with. A kept run whose own search moves it by at most two grid steps is moved by that amount, which 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 10 considers moving it up to $R = 60$ cues either way, never past the boundaries on either side of it, and evaluates

$$J(j) = \sum_{k=i-R}^{j-1} \psi_k(b) + \sum_{k=j}^{i+R} \psi_k(a) + \tfrac{1}{2}\,\mathbf{1}[\text{a chapter mark lies at } j], \tag{45}$$

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

Proposition 9.4. *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. Without the chapter term, $J(j+1) - J(j) = \psi_j(b) - \psi_j(a)$, 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 chapter term is a function of j alone and is added when j is compared. 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 window stops at the neighbouring boundaries because the split search sometimes leaves a few boundaries close together, and without that limit the first one can slide straight past the others and fill everything it passes over with its own two offsets, which wipes them out.

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 withut ruling it out.

Algorithm 10. Settling one boundary

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

$J \leftarrow \sum_{k=i-R}^{i-1} \psi_k(b) + \sum_{k=i}^{i+R} \psi_k(a)$, both sums clipped at the neighbouring boundaries
best $\leftarrow J + \frac{1}{2}$[chapter at i]; at $\leftarrow i$; widest $\leftarrow c_1^i - c_2^{i-1}$
for $j = i+1, \dots, i+R$ **do** ▷ walk right
  $J \leftarrow J + \psi_{j-1}(b) - \psi_{j-1}(a)$
  if $J + \frac{1}{2}$[chapter at j] beats best, or ties it with $c_1^j - c_2^{j-1} >$ widest **then** update best, at, widest
  end if
end for
walk left from $i-1$ to $i-R$ in the same way, with $J \leftarrow J + \psi_j(a) - \psi_j(b)$
give cues before at the offset b and cues from at onwards the offset a

9.5 Chapters

A television recording puts its chapter marks on the advert breaks, and that is where a subtitle from another release needs to jump. Sushi [6] takes chapters as the likely places a release was cut, and LAPSE does too, in two places. The split search charges $p/2$ for a change where a chapter mark of the reference falls between the two cues, within 2 s, and the boundary step counts a chapter as half a cue. The second does most of the work: a boundary in a stretch with little speech otherwise goes wherever the score tips. With a 4 s break added at a chapter mark 15 minutes into a test film, the three lines before the break that go the wrong way without the hint stay put. Chapters every five minutes on a film that was not cut change nothing, and so do chapters that are not on the break.

If the penalty is read as a negative log prior on a change at each cue boundary, which is a loose reading (Section 9.6), halving it at a chapter multiplies the prior odds of a change there by $e^{p/2}$. With $p = \ln N$ that is $\sqrt{N}$, about 39 for a film of 1 500 cues.

9.6 The penalty

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

$$p = \max\bigl(3,\ \ln N\bigr). \tag{46}$$

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 $|\Sigma|$ 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 [27], 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. The free jump at $2p$ reads the same way: a missing scene is two changes, one where the scene would start and one where the subtitle resumes, with nothing to score in between.

9.7 When a split is kept

Not every split the recursion returns is worth having. The shifted and drifting paths of Section 10 run the fine grid ($W = 45$ s) and keep its result only if it passes a check:

Definition 9.5 (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 a reach ϱ of σ_0, and the largest and smallest offsets differ by at least 120 ms. The reach is 60 s for the fine grid and 600 s for the wider search of Section 10.5.

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 recut path of Section 10 does not apply the check. The check would not remove runs a few tens of milliseconds apart either, once the answer as a whole spans more than 120 ms, so an answer in pieces can contain them; the worked example has two (Section 11.4).

9.8 Restarts and joined parts

Two structures look like splits and are cheaper to find 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, provided both sides are real parts: at least $\max(30, N/10)$ cues since the previous restart, and the cue that many places further on still before the one where the jump happened. Each part is aligned on its own within ±90 minutes, and the file then goes through the recut search with the first part's offset as anchor, which moves the boundary to where it pays and catches a jump inside a part. 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, and with the parts further apart than any recut would move them. Sixteen slices are aligned on their own within ±90 minutes. Among the backed slices whose confidence times margin is at least 0.02, an offset more than five minutes from the offset of the current part marks a new part. For each pair of neighbouring parts with offsets σ_a and σ_b, the cut is placed at

$$j^\star = \arg\max_j \Bigl(S_{<j}(\sigma_a) + S_{\geq j}(\sigma_b)\Bigr), \tag{47}$$

where $S_{<j}$ scores the cues before j and $S_{\geq j}$ the rest. Each cue is scored once at either offset and the two sums are read off running totals, so the scan costs $O(N \log K)$.[7] Each part then gets its own offset, and the file goes through the recut search like a restart.

[7] An early version copied and rescored both halves for every candidate cut, which was quadratic.

Algorithm 11. Joined parts

Input: cues c in file order, reference
$(\sigma_1..\sigma_{16}) \leftarrow$ offsets of 16 slices, each within ±90 min
anchors $\leftarrow$ offsets of backed, confident slices more than 5 min from the last anchor
if fewer than two anchors **then return** nothing
end if
for consecutive anchors (σ_a, σ_b) **do**
    $j^\star \leftarrow \arg\max_j S_{<j}(\sigma_a) + S_{\geq j}(\sigma_b)$ over cues after the previous cut, by running totals
end for
align each part on its own; lock $\leftarrow$ lowest part lock
SplitAlignment(anchor = offset of the first part, $W = 600$ s) ▷ Algorithm 9

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. Merging overlapping cues in split mode welded the two halves of a cut file together, so for a while split mode kept the cues exactly as they were in the file; in 2.3.1 that became the present rule, which glues only a cue that starts inside the one before it. The fixed penalty of 6 became $\max(3, \ln N)$ in 2.0.0, and the boundary settling and run refinement came in the same release. Version 2.3.0 replaced the regular grid by the irregular one, added the free jump and the chapter hints, and stopped boundaries from passing each other.
Without the free jump, a test film with 40 s cut ten minutes in had the 150 cues before the cut put 40 s out by both auto and split, and both called the result solid. The islands at first took every slice offset, and a file with three advert breaks of 30 s came out with pieces at −1 844, −5 070 and +463 s; that is where the second slice came from. The requirement then went too far the other way. A subtitle whose last four minutes sat 20 minutes out was split with that piece at +460 s, because only one slice covered the piece, at +1200 s and $\Lambda = 14.7$, the clearest lock of any slice in the file, and an offset that needed two slices to agree never reached the grid. The last changes added the check on runs and let a single clear slice back a far offset.

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, and 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, one per five cues, and none below 10), and Algorithm 6 is run on each slice on its own within ±15 minutes. The result is eight points (t_j, σ_j) with t_j the mean midpoint of the slice, each with its own lock Λ_j. Three summaries are taken from them, all with the tolerance $\eta = 400$ ms:

$$F = \#\{j : |\sigma_j - \hat{\sigma}| \leq \eta\} \quad \text{slices that back the global offset,} \tag{48}$$

$$L = \#\{j : |\sigma_j - \hat{\varepsilon} t_j - \hat{\sigma}_0| \leq \eta\} \quad \text{slices on the Theil and Sen line,} \tag{49}$$

$$C = \text{number of groups of at least two slices} \quad \text{with consecutive offsets within } \eta. \tag{50}$$

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

10.2 What agreement is worth

Three agreeing slices are hard to get by chance, and the following bound puts a number on it.

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 $P(\geq 3) \approx \binom{8}{3} q^3 \approx 56 \times 8.8 \times 10^{-11} \approx 4.9 \times 10^{-9}$. □

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 Looking for a far part

A part of the file that sits far from the rest can hide from the eight slices. If one slice mostly covers it, only one slice disagrees, and the slices beside it are outweighed by their own cues. If it is more than 15 minutes away, no slice can reach it. So auto mode looks for such a part on every file.[8] A *second part* is present when two slices, each more than 45 s from the global offset and each with $\Lambda_j \geq 6$, lie within η of each other. The eight slices are checked first, and if they show nothing, the file is cut into sixteen slices aligned within ±90 minutes and those are checked.

Under the independence assumption of Proposition 10.1, two of sixteen wrong slices land within 400 ms of each other somewhere in ±90 minutes with probability at most $\binom{16}{2} \cdot 801/10\,800\,001 \approx 8.9 \times 10^{-3}$, before the requirement $\Lambda_j \geq 6$ on each is taken into account. In development the sixteen slices made a run on a 70 minute film whose slices all agree about 1.5 s longer; across the development films the cost was 0.4 to 1.9 s. On the benchmark machine a whole cached run took a median of 0.71 s against 0.66 s for 2.0.0 (Section 12.8).

10.4 The decision

Algorithm 12 is the decision procedure. The order of the tests matters: a file is called shifted if it can be, and drifting only if the slices lie on a line better than they lie on a level.

Algorithm 12. Mode selection in `auto`

Input: subtitle cues, reference
if the file restarts (Section 9.8) **then return** RESTART
end if
$(\hat{\sigma}, \Lambda_1) \leftarrow$ BESTOFFSET(c); slices ← eight slice offsets; compute F, L, C, $\hat{\varepsilon}$
drifts ← $L \geq 4$ and $|\hat{\varepsilon}| > 10^{-5}$; *far* ← second part among the eight slices
if $F \geq 3$ and not (*drifts* and $L \geq F + 2$) and not LOOKFAR **then** kind ← SHIFTED
else if *drifts* **then** kind ← DRIFTING; $\rho \leftarrow$ snap($1 + \hat{\varepsilon}$), or Algorithm 8 on $1 + \hat{\varepsilon} \pm 0.003$ if it beats that by 1
else if $C \geq 2$ or *far* **then** kind ← RECUT
else
  $(\rho, \Lambda_\rho) \leftarrow$ RATIOSEARCH ▷ Algorithm 7
  if $\rho \neq 1$, $\Lambda_\rho \geq \theta$ and $\Lambda_\rho > \Lambda_1 + 2$ **then** kind ← DRIFTING
  else if LOOKFAR **then** kind ← RECUT
  else $(\rho, \Lambda_\rho) \leftarrow$ Algorithm 8 on $[0.94, 1.06]$; DRIFTING under the same test, else SHIFTED
  end if
end if
if RECUT **then return** JOINED if Algorithm 11 finds parts, else SPLITALIGNMENT($\hat{\sigma}$, 600 s, 100 ms)
end if
if DRIFTING **then**
  choose ρ among the ratios of P within 0.004 by Λ after the fine split ▷ Section 8.6
  $(\sigma_\rho, \Lambda_\rho) \leftarrow$ BESTOFFSET(ρc)
  if $\Lambda_\rho < \Lambda_1 + 1$ **then** continue as RECUT if LOOKFAR found a part, else as SHIFTED

[8] Without this search, a subtitle with a six or ten minute part 18 or 20 minutes out was called solid at one offset and written over with that part in the wrong place.

```
    end if
    verdict from Λ_ρ and eight new slices of ρc
    if the verdict is not nothing then
        a second part in sixteen slices of ρc within ±90 min ⇒ recut search on ρc (DRIFTING+RECUT)
        otherwise fine split of ρc; if worth splitting,
            REACHFURTHER and judge the pieces (DRIFTING+SPLIT)
    end if
end if
if SHIFTED then
    verdict from Λ_1 and F
    if the verdict is not nothing then
        fine split, W = 45 s, δ = 25 ms; if worth splitting,
            REACHFURTHER and judge the pieces (SHIFTED+SPLIT)
    end if
end if
return kind, offsets, verdict
```

The reported mode names the path: `auto/shifted`, `auto/shifted+split`, `auto/drifting`, `auto/drifting+split`, `auto/drifting+recut`, `auto/recut`, `auto/joined` and `auto/restart`. The split penalty is set from the file size (Section 9.6) in every path.

Two differences from the procedure of 2.0.0 account for most of the improvement in Section 12.7. In 2.0.0 the fine split only ran after a *solid* verdict on the global offset, and whatever it found was kept or dropped by Definition 9.5 alone; nothing measured whether the split made the file better. A cut keeps any one offset from standing out, so a file cut in two by 2.6 s came back *unsure* with one offset for all of it. The fine split now runs for an unsure answer as well, and a split answer is judged on the file it leaves behind. And in 2.0.0 a cut whose two sides were more than 45 s apart could only be found through the recut path, which needed the slices to fall into groups first; the far part search and Section 10.5 give it two more ways in.

10.5 Reaching further

When at least three of the eight slices agree on one offset and no far part turns up, auto takes the shifted path, and the fine split that follows only looks 45 s each way. A part further out than that is pressed against the edge of the window, where some of its lines happen to overlap speech.[9] A fine split that is worth keeping is therefore checked twice more. If some piece lies within a tenth of the window of its edge, $|o - \sigma_0| \geq 0.9W$, or if the slices contradict the pieces in the sense of Definition 10.3, the recut search runs over the whole irregular grid with the same anchor. Its result replaces the fine one if it is worth splitting with a reach of 600 s and leaves the file with a higher Λ once the pieces are moved. On the development films the five minute part then lands within 95 ms, with or without a drift on top.

10.6 The verdict

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

$$\text{verdict} = \begin{cases} \textit{nothing} & \text{if } \Lambda < \min(6, \theta), \\ \textit{solid} & \text{if } \Lambda \geq \theta \text{ and } A \geq 3, \\ \textit{unsure} & \text{otherwise.} \end{cases} \tag{51}$$

For a single offset, Λ is the lock of that offset and A is F; for a stretch, they are the lock of the stretched file and the number of its eight new slices that agree with the stretched offset. For an answer in pieces, from the split, recut and the two `+split` paths, both are computed on the finished file. Move

[9] A film with its last five minutes 52 s out once came back solid with that part moved by 43.7 s. On the development films that five minute part was 8 and 50 s out, with and without a drift on top, before the checks described here.

every cue by its piece's offset, sort, and let

$$\Lambda_{\text{after}} = \Lambda\big(\textsc{BestOffset}(\text{moved cues})\big), \tag{52}$$

which for a correct answer is the lock of an offset close to zero. A is the number of slices whose own offset lies within η of the piece that covers the slice's midpoint. A third condition then guards against a wrong piece.

Definition 10.3 (Contradicted)**.** A set of piece offsets $o_1, \dots, o_q$ is contradicted by a slice with offset σ_j and lock $\Lambda_j \geq 4$ if $\min_p |\sigma_j - o_p| > 2\,\text{s}$, and either $\min_p |\sigma_j - o_p| \leq 45\,\text{s}$ or $\Lambda_j \geq 5$. Only slices that can reach their piece count: a slice whose expected offset, the offset of the piece at its midpoint, lies beyond ±15 minutes is left out.

A *solid* verdict on pieces that are contradicted becomes *unsure*. The lower threshold of 4 for a slice within 45 s of a piece is what a missed advert break looks like: a slice that locks on modestly a few tens of seconds from every piece.

The reach condition keeps a slice from objecting to a piece it could never have seen. Without it, a subtitle whose second half sat 20 minutes out came back split correctly and *unsure*, because the slices over that half, which only look 15 minutes either way, locked on somewhere else at Λ between 5 and 9 and held the answer back. A slice that could have found its piece and did not still counts, and that is what holds back the wrong answers: a drifting file with a tail 20 minutes out, which auto split into −0.1 and +14.9 s, and a 66 minute film with its last four minutes 18 minutes out. Their Λ and confidence were the same as on correct answers, so no threshold on either would have told them apart.

The restart and joined paths use the lowest lock among their parts as Λ, 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 no sidecar, and `--no-sidecar` writes nothing either way; both of the last two exit with 3 as the table says.[10] 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.

10.7 From one release to another

With `--from`, the subtitle is known to fit a second video, and the two soundtracks say exactly how the copies differ, which is far more to go on than a subtitle against speech. The speech of the second video is found and cached the same way as the reference's, the lead-in is applied to both so that the two cancel, and its spans take the place of the cues in everything Algorithm 12 does. The answer is a correction per speech span. Each real cue then takes the correction of the speech span nearest its start, the span that starts at or before it unless the next span's start is closer than the end of that one, and snapping and writing work on the real cues as before. In development, a 45 minute test film played back 4.27% faster had every line land within 10 ms this way, where the same subtitle synced against the speech was 145 ms out. Nothing in the benchmark measures `--from`.

[10] They used to exit with the code for a file with too few cues.

By hand. `--shift` moves every line by a number of milliseconds, and `--ratio` stretches every time first, so that a line at t lands at $\rho t + \sigma$. Nothing is measured, so the verdict is *solid* by definition, and a shift of zero with a ratio of one leaves the file alone.

10.8 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 13 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. The end of a line is also pulled onto a cut near it, as long as the line stays at least 500 ms long and does not run into the next line.

Algorithm 13. Snapping cue starts and ends

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
if at least three gaps lie within 80 ms of the longest **then** ▷ the encoder's maximum GOP
  drop the keyframes that end such a gap
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
for each cue $[s, e)$ **do**
  $\kappa \leftarrow$ the keyframe nearest to e
  if $0 < |\kappa - e| \le \omega$, $\kappa - s \ge 500$ ms, and no other cue starts between e and κ **then** $e \leftarrow \kappa$
  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; engine then leaves the file alone and says so. When a few gaps sit at the maximum length the encoder allows, the keyframes at their ends were forced by that limit and are not cuts, and they are dropped. Snapping is off by default, runs only against a video reference, and never changes how the offset itself is found. The start pass does not re-check the order of neighbouring cues, 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 point is to watch the mechanism work. It says nothing about accuracy, which is the subject of Section 12: the starting subtitle is a copy that its owner corrected by hand against this release, and nobody outside that process has checked it, so the example measures every result against the injected error.

The film is *(500) Days of Summer* (2009): 95 minutes, an MP4 file with AC-3 stereo audio at 48 kHz, and an English SDH subtitle of 1 434 cues. Three broken copies were made from the subtitle:

- **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 that start before 00:48:00 moved 21 s early and the rest 33 s late, a cut with a spread of 54 s.

The runs used LAPSE 2.5.0 on the Mac mini with an M2 Pro that also ran the benchmark, in the default `auto` mode with `--no-embedded`. The intermediate values come from a build of the same code with extra logging switched on by an environment variable; the subtitle it wrote was byte for byte the same with the logging on and off.

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 32 pieces of three minutes, eight at a time, and Silero found 1 331 speech spans, the first of them 50.85 s into the film. The first run took 5.9 s of wall time and 22.7 s of user CPU time across the cores. The spans were cached. Later runs on this film took about 1.6 s for the shifted copy, 3.5 s for the drifting one and 3.2 s for the cut one, of which a good part is the search over sixteen slices to ±90 minutes that Section 10.3 runs on every file.

11.2 The shifted copy

FFT search. The reference occupies $n_r = 139\,130$ bins of 40 ms, so the close plan has $N_{\text{fft}} = 262\,144$. 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	−33 000	43.0
2	both, 550 ms	−33 000	30.9
3	onsets, 450 ms	−32 920	30.3
4	blocks, 250 ms	−33 120	28.8

The strongest rival in any pass was −30 640 ms with $z = 8.6$ in pass 4, 2.4 s from the answer: the same dialogue lined up one line off. After pooling and thinning, 21 candidates remained, offset zero among them.

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
−33 000	17.19	−33 087	590.98	**16.74**
−35 520	7.96	−34 314	529.41	4.06
−30 640	6.45	−31 846	520.86	6.01
−20 560	4.85	−21 227	456.07	4.47

Offset zero came sixth with $\Lambda = 3.03$ and was not refined. The two rivals at −34.3 and −31.8 s score 10 and 12% less on S than the winner, and their Λ is a quarter and a third of it. The raw score separates the right answer from a neighbouring line by a tenth; the lock statistic separates them by factors of 2.8 and 4.1.

The lock of the winner. At −33 087 ms the bunching was $B = 0.509$ against a random baseline of $\mu_B = 0.236$, $s_B = 0.020$, so $z_B = 14.0$; the overlap per cue was $O = 0.431$ against $\mu_O = 0.276$, $s_O = 0.024$, so $z_O = 6.6$. The whole file term is $0.6\,(14.0 + 6.6) = 12.36$. The eight slice terms were 2.06, 3, 1.88, 3, 3, 3, 3, 2.94, five of them at the clip of 3, with a mean of 2.74 and a contribution of $1.6 \times 2.74 = 4.38$. Together $\Lambda = 16.74$. The confidence, S/N, was 0.431.

Slices and decision. Aligned on its own, each slice of 171 cues (174 in the last) gave:

Slice	Midpoint	Offset (ms)	Λ_j
1	7:02	−33 209	4.6
2	16:58	−33 111	7.6
3	27:22	−33 145	3.6
4	38:59	−33 071	7.6
5	50:55	−33 098	13.0
6	1:02:44	−33 079	8.8
7	1:13:09	−32 998	7.3
8	1:27:15	−33 007	6.3

All eight lie within 122 ms of the global offset, so $F = 8$. The search for a far part cut the file into sixteen slices aligned within ±90 minutes. Twelve found the offset again, and the other four landed at +1 076, +433, −1 850 and −4 214 s, with locks between 3.4 and 5.9, no two of them close: four slices locked onto nothing in particular, which is what quiet slices do. There was no second part, the file is SHIFTED, and with $\Lambda = 16.7 \geq 8$ and eight agreeing slices the verdict is *solid*. The fine split search then ran with $W = 45$ s, $\delta = 25$ ms and $|\Sigma| = 3\,601$, and returned a single run.

Result. Offset −33 087 ms, written over the file. That is 87 ms more than the 33 s that were injected, the same on all 1 434 cues, so the corrected file sits 87 ms earlier than the copy it was made from. The difference reappears as 86 ms in the drift run below. It is a constant difference between this subtitle's timing and the engine's reading on the audio, lead-in included; which of the two is closer to the speech is a question the example cannot answer.

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 −53 447 ms with $\Lambda = 3.29$. The slices scattered over ±15 minutes with locks between 3.2 and 4.4, none agreed with the global offset, one lay on the Theil and Sen line, and they formed no groups. The engine fell through to the ratio search of Algorithm 7:

Ratio ρ		Offset (ms)	Λ
1		−53 447	3.29
24/23.976, 30/29.97	1.00100	−275 928	3.09
23.976/24, 29.97/30	0.99900	−101 088	2.78
25/24	1.04167	−327 953	2.91
24/25	0.96000	−526	8.90
25/23.976	1.04271	−202 342	2.83
23.976/25	**0.95904**	**−86**	**17.53**
30/25	1.20000	−394 032	2.57
25/30	0.83333	−306 099	3.21

The correct ratio stands out with $\Lambda = 17.5$, and so does its neighbour: 24/25 differs from it by 0.1%, stretches the file by the right amount to within 5.5 s over the whole film, and reaches $\Lambda = 8.9$, just over the solid threshold and about half the lock of the correct ratio. Figure 5 shows the same thing at a finer scale. The ratio passed both tests of Algorithm 12: $17.5 \geq 8$ and $17.5 > 3.29 + 2$.

Because the file was now called drifting, the comparison of Section 8.6 ran. 24/25 is within 0.004 of 0.95904, so both were tried with the fine split: 23.976/25 needed no split and left the file at $\Lambda = 17.53$, while 24/25 was cut into a staircase of three pieces at −0.99, −2.89 and −5.12 s that reached 11.64. The engine kept 23.976/25, searched the scaled file again, found −86 ms with $\Lambda = 17.5$, which beats the unscaled lock by far more than one unit, and aligned eight new slices of the scaled file. All eight agreed. The search for a far part in the scaled file found nothing, the verdict was *solid*, and the fine split found nothing either.

Result. Ratio 0.95904, offset −86 ms, reported as `auto/drifting`.

11.4 The cut copy

Slices. Slices 1 to 4, before the cut, found offsets between +20 791 and +20 929 ms. Slices 5 to 8, after it, found between −33 098 and −32 998 ms. The slices 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, −33 040 ms reached $\Lambda = 10.78$ and +20 878 ms reached 10.64: two nearly equal locks for two parts of the file. With −33 040 as the global offset, $F = 4$, which would allow the shifted path, and the far part test of Section 10.3 is what keeps the file off it. Slices 2 and 4 lie 54 s from the global offset, within 40 ms of each other, with locks of 7.8 and 7.9: a second part. The file went to the recut path. The jump of 54 s is far below the five minutes that mark joined parts, so the recut search ran around −33 040 on the plain grid of 12 001 offsets, with no islands, since no slice offset outside the window was backed.

Result. Four pieces, at +20 859, +20 929, −33 080 and −32 998 ms, with $\Lambda_{\text{after}} = 17.68$ against 10.78 for the best single offset, all eight slices in agreement with the pieces, and the verdict *solid*. The forced `split` mode returned exactly the same pieces in 2.3 s against 3.2 s for auto.

Two things in this result need a comment. First, two of the three boundaries are not cuts. The steps from +20 859 to +20 929 and from −33 080 to −32 998 are 70 and 82 ms, inside each half. Nothing in the recut path merges neighbouring runs that close together, and Definition 9.5 would keep them as well, since its 120 ms rule only asks that the largest and smallest offsets of the whole answer differ by that much. They are harmless at the scale of a subtitle and they make the JSON line say four parts where there are two.

Second, the boundary at the real cut is four lines early. The cut was injected at 48:00, between a doorbell at 47:59.6 and a sigh at 48:03.6 in the original timing. The engine put it before the line “After you, please.” at 47:17.9, so that line and the three sound descriptions after it, which follow the line before them in the file, were given the offset of the second half and are 54 s out. In the corrected file they now come before the line in front of them, by up to 48 s. This is the case that Section 9.4 warned about: the boundary settling moves a boundary to where the score is best without checking the order of the cues, and “After you, please.” is the only scored line between 47:11 and 48:34, so a small gain at the other offset was enough to move it.

Both are on the list of loose ends in Section 14.4. The second is the more annoying one in practice, since four lines out of order by most of a minute is something a viewer actually sees.

12 Evaluation

All numbers in this section come from `docs/benchmarks.md` as published on 13 November 2026 (commit `9ef92ea`), which measured LAPSE 2.5.0 and 2.0.0 against alass 2.0.0 and ffsubsync 0.5.1 on the same 39 films. Every film was run again on 2.5.0. alass and ffsubsync keep their results and times from August and were not run again. The per film tables in Appendix B reproduce the source, and every count below is computed from its per film rows by the script in `tools/make_tables.py`.[11]

12.1 Questions

The benchmark was built to answer four questions, and this edition adds a fifth:

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

[11] The release of the 2.5.0 numbers also corrected three totals for the other tools that did not match their own rows in the August version: alass 28 and ffsubsync’s split mode 18 in the split test, and 1 for ffsubsync without splitting. The previous edition of this report had already used the per film rows. The totals of the times quoted here are summed from the per film rows and can differ from the summary in the source in the last digit; the source gives 34.8 s for the cached runs of 2.5.0, the rows sum to 34.85.

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?

Q5 What changed between 2.0.0 and 2.5.0 on the same films?

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 synchronisation hard: 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`. LAPSE ran with `--no-embedded` on every film, so its runs were aligned against the audio, like those of the other tools. ffsubsync is much faster on the four `.mkv` files than on the rest, so its times there should not be compared with the others.

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 among the twelve films of 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.[12]

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.

12.4 Tools and configurations

Run	Configuration
LAPSE auto	default mode, penalty chosen by the engine; in the normal test run twice, the second run reading the speech cache
LAPSE forced	`split` mode, penalty chosen by the engine, cached
alass	as shipped; splitting is on by default, penalty 7
ffsubsync	as shipped; no splitting
ffsubsync split	with `--split-penalty` and no value, which gives the default penalty of 5

All runs were single runs on a Mac mini with an M2 Pro and nothing else running. 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] Two rows of the source table are incomplete. *Apocalypse Now* is listed with two cuts and three segment offsets and only one cut time, and *Badlands* with three cuts and four offsets and two cut times. The offsets, and so the spreads used in Section 12.10, are complete.

12.5 Scoring and statistics

I checked every output of all three tools by hand, watching the film. A run *passes* when the subtitles are in sync to the point where a viewer does not notice an offset, and is *partial* when some segments land and others do not, as long as at least half do; below half it fails. Scoring used a soft limit of about 250 ms. A run with cues beyond 250 ms could still pass if the subtitles looked right on watching, and a run inside 250 ms could fail if they looked wrong, for any of the three tools. The same soft limit applied to all three, so the comparison is fair. 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.

With 39 films, a difference of a few passes can be noise.

Pass rates are reported with 95% Wilson score intervals [34], which behave well at rates near 1 where the textbook interval does not [1]. Differences between two runs on the same films are tested on the paired outcomes with an exact McNemar test [20]: if b films pass only with the first and c only with the second, the two sided p value is $\min\bigl(1,\ 2\sum_{k\le\min(b,c)}\binom{b+c}{k}2^{-(b+c)}\bigr)$. Partial results count as failures in these tests. No correction for multiple comparisons is applied. Ten comparisons are reported, and with a Bonferroni correction at the 5% level only those below 0.005 would stand; the text says which those are.

12.6 Shift and drift

	LAPSE 2.5.0	LAPSE 2.0.0	alass 2.0.0	ffsubsync 0.5.1
Passed, of 39	**37**	36	31	33
95% interval	83 to 99 %	80 to 97 %	64 to 89 %	70 to 93 %

Table 4. Normal test (shift or drift).

LAPSE 2.5.0 failed two films, *The Godfather* and *The Godfather Part II*, which failed for every tool in both versions. It passed *Stalker*, the one film 2.0.0 had failed in this test and alass had passed. With 2.0.0, it was the only version of any tool to pass *Come and See* (alass crashed after 0.09 s, ffsubsync landed 33 s off) and *The Godfather Part III*.

On the paired outcomes, 2.5.0 passed 6 films that alass failed and alass passed 0 that 2.5.0 failed ($p = 0.031$). Against ffsubsync the split is 4 to 0 ($p = 0.125$), and against 2.0.0 it is 1 to 0. The lead over alass is significant at the 5% level, which it was not for 2.0.0 in the previous edition ($p = 0.125$), and it would not survive a correction for the ten comparisons. No film that any other tool passed was failed by 2.5.0.

12.7 Cuts

	LAPSE 2.5.0		LAPSE 2.0.0		
	auto	forced	auto	forced	alass
Passed, of 39	34 (+2)	**37** (+1)	19 (+5)	32 (+3)	28 (+1)
95% interval	73 to 94 %	83 to 99 %	34 to 64 %	67 to 91 %	56 to 83 %

Table 5. Split test. Partial results in brackets. ffsubsync passed 18 (+1) with `--split-penalty`, interval 32 to 61 %, and 1 (+2) without it.

With splitting forced on, 2.5.0 passed 37 films against 28 for alass (9 to 0 on the paired outcomes, $p = 0.004$) and 18 for ffsubsync's split mode (19 to 0, $p = 3.8 \times 10^{-6}$). Both differences are clear and survive the correction. Against forced 2.0.0 it gained five films and lost none ($p = 0.063$): *Enter the Void*, *Manchester by the Sea*, *Seven Samurai*, *The Deer Hunter* and *The Hunt*.

The change that matters most for a library is in the automatic column. Left to decide by itself, 2.0.0 passed 19 films and 2.5.0 passes 34. On the paired outcomes 2.5.0 gained 15 films and lost 0 ($p = 6.1{\times}10^{-5}$,

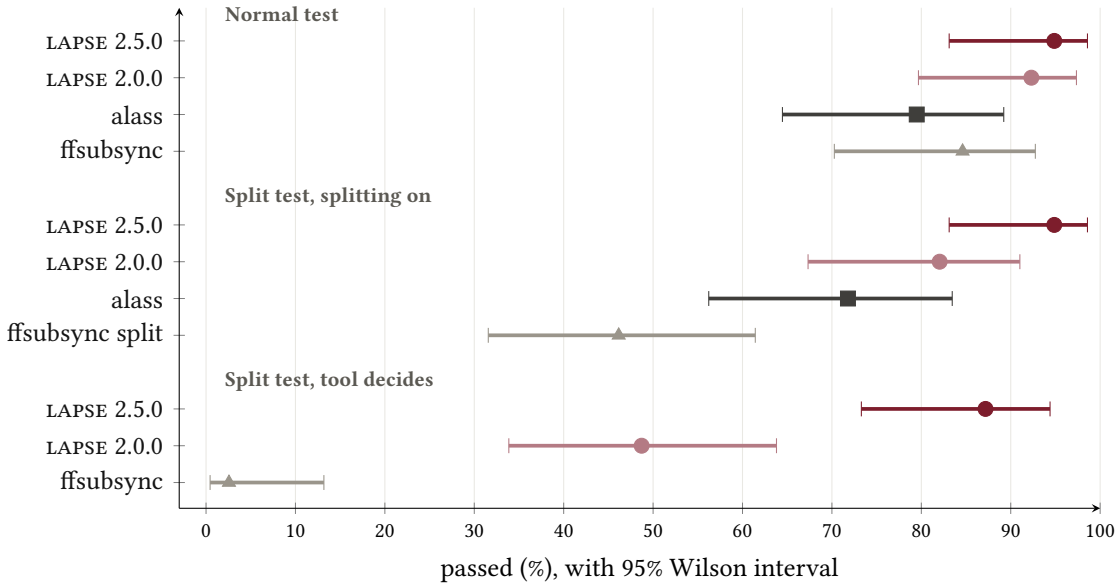


Figure 6. Pass rates on 39 films with 95% Wilson intervals. In the bottom group alass has no entry because it always splits, and ffsubsync is shown without splitting, which is what it does unless told otherwise.

which survives the correction). The gap between deciding and being told has nearly closed: in 2.0.0, forcing the split helped on 14 films and hurt on 1 ($p = 9.8 \times 10^{-4}$, which survives it as well); in 2.5.0 it helps on 3 and hurts on none ($p = 0.250$). The three films that still need the split forced are *Manchester by the Sea*, *The Godfather* and *The Godfather Part III*, and Section 12.10 looks at why.

Table 6 breaks the results down by the number of cuts. With one cut, 2.5.0 passes 22 of 25 by itself and 24 when forced. With three or four cuts it passes all six films in both modes, as alass does. The groups with more cuts are small.

Cuts	Films	LAPSE 2.5.0 auto	LAPSE 2.5.0 forced	LAPSE 2.0.0 auto	alass	ffsubsync split
1	25	22	24	17	18	13
2	8	6 (+2)	7 (+1)	0 (+4)	4 (+1)	4 (+1)
3	5	5	5	2 (+1)	5	1
4	1	1	1	0	1	0

Table 6. Split test by number of cuts: passes, with partial results in brackets.

For the twelve films of the first round, the source also records what each tool did at the cut. 2.5.0 made one cut on the right place on 7 of the 12, one cut elsewhere on 4, and several cuts on one, *Stalker*, with six. 2.0.0 had made one clean cut on 5 and had answered *Stalker* with 102 cuts, *Saving Private Ryan* with 38 and *Uncut Gems* with 15. alass made one clean cut on 5 and several cuts on 5, among them 98 on *Requiem for a Dream* and 101 on *Stalker*; ffsubsync's split mode made one clean cut on one film. Every extra cut is a place where the subtitle jumps during playback, so these counts matter beyond the pass mark: a file that passes with a hundred cuts is unpleasent to watch.

12.8 Time

Figure 7 shows every normal run on a logarithmic axis. Without the cache, 2.5.0 took a median of 8.82 s per film (interquartile range 6.48 to 11.87 s), 2.0.0 10.82 s, alass 11.59 s and ffsubsync 17.25 s. 2.5.0 was faster than 2.0.0 on all 39 films, with a median ratio of 0.81, and faster than alass on 30 of 38, with a median ratio of 0.64. The totals over 39 films were 373.27 s for 2.5.0, 511.15 s for 2.0.0, 514.14 s for alass and 656.27 s for ffsubsync. The gain over 2.0.0 comes from the decoding of Section 5.1: shorter pieces that keep every core busy, the shorter resampling filter and the channel map.

With the cache the order reverses. 2.5.0 took a median of 0.71 s (0.56 to 1.08 s) against 0.66 s for 2.0.0, slower on 36 of 39 films, with a median ratio of 1.19 and totals of 34.85 against 29.66 s. The likeliest source of the extra time is the search for a far part over sixteen slices within ±90 minutes (Section 10.3), which runs on every file. In development that search added 0.4 to 1.9 s to a run, several times the median difference here. I have not found what accounts for the gap between those two measurements.

A cached run is still a median 11.3 times faster than the first run on the same film and 13.4 times faster than alass, which was also run twice and took about as long the second time.

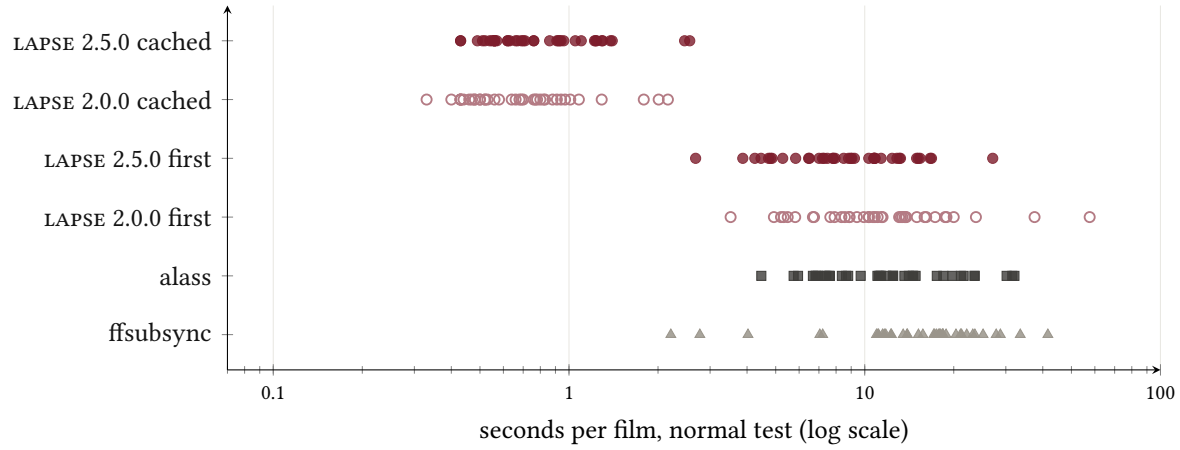


Figure 7. Time per film in the normal test, one mark per film. alass's crash on *Come and See* is left out. The ffsubsync runs between 2 and 7 s are the `.mkv` films, where the container changes its timing.

In the split test all LAPSE runs were cached. The medians were 0.77 s for auto and 0.74 s for forced splitting in 2.5.0, 0.72 s and 0.62 s in 2.0.0, against 12.32 s for alass and 20.94 s for ffsubsync's split mode. The split recursion adds little to a run.

12.9 The verdict

	Normal			Split, auto			Split, forced		
Verdict	pass	part	fail	pass	part	fail	pass	part	fail
solid	22	0	0	16	0	0	17	0	0
unsure	2	0	2	5	2	2	8	1	1
nothing	1	0	0	1	0	1	0	0	0

Table 7. Verdict of LAPSE 2.5.0 against outcome for the 27 films of the second round.

Table 7 crosses the verdict with the outcome. Every *solid* answer was right in all three tests: 22 of 22 in the normal test (95% interval for the precision 85 to 100 %), 16 of 16 in auto (81 to 100 %) and 17 of 17 forced (82 to 100 %). The three columns are the same 27 films, so the 55 answers cannot be pooled into one interval as if they were independent.

The auto column is where 2.5.0 differs most from 2.0.0. There, 2.0.0 had called 16 answers *solid*, of which 7 passed, 3 were partial and 6 failed (23 to 67 %). Its verdict judged the global offset before the split ran, and the split result was never judged. The engine computes its verdict on the finished file (Section 10.6), and none of the 16 *solid* answers in auto is wrong.

The cost is what the verdict holds back. In the normal test, 3 correct answers went to a sidecar, as in 2.0.0. In auto it is 6, against 2 in 2.0.0, and forced 8, against 6. Some of that is the contradiction rule doing its job too eagerly: *Zoolander*, *Possession* and *Apocalypse Now* were *solid* and correct in auto and *unsure* and correct when forced, where the slices cannot see the split that forcing made.

Two answers were *nothing* and right: *Manchester by the Sea* in the normal test and *Blue Velvet* in auto. A right answer behind a *nothing* is the least useful outcome, since a user has the least reason to

look at it. *Manchester by the Sea* is *nothing* in the normal test, *nothing* and wrong in auto, and *solid* and right when forced, the same pattern as in 2.0.0 except that the forced answer is now right.

12.10 Spread and automatic splitting

The previous edition predicted which films auto mode should fail: the fine split only looked ±45 s around the global offset and Definition 9.5 rejected any piece further than 60 s from it, so a file whose segment offsets spread more than 45 s could only be fixed through the recut path. Take the *spread* of a film as the difference between its largest and smallest injected segment offset. Figure 8 plots the spread against the outcome in auto mode for both versions.

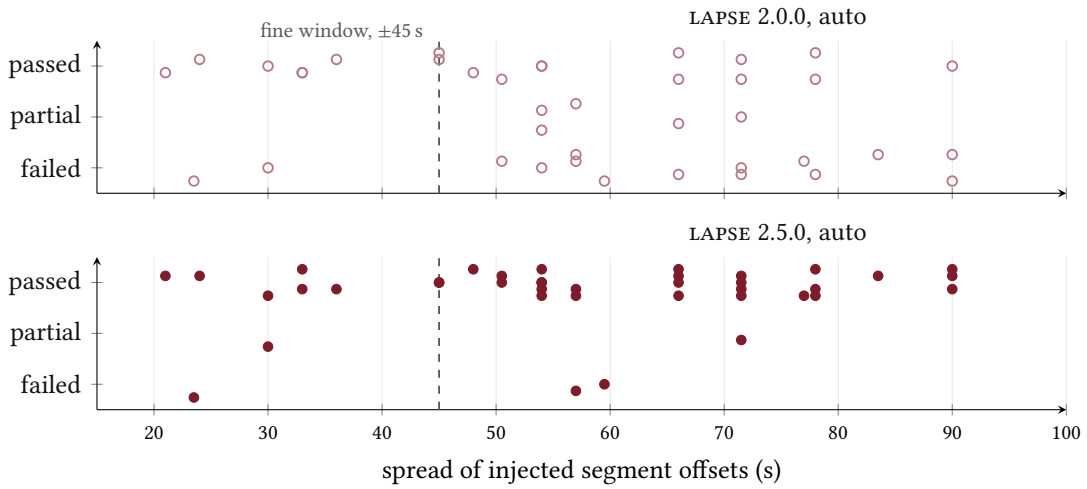


Figure 8. Outcome in auto mode against the spread of the injected offsets, all 39 films. Marks are offset vertically by a small fixed amount so that films with the same spread stay visible.

For 2.0.0 the prediction holds. Of the 10 films with a spread of 45 s or less, it passed 8 and failed 2. Of the 29 with a larger spread, it passed 11, was partial on 5 and failed 13. In 2.5.0 the dependence on the spread has nearly gone: 8 passed, 1 was partial and 1 failed below 45 s, and 26 passed, 1 was partial and 2 failed above it.

When the spread is above 45 s, a segment can lie more than 45 s from the global offset, outside the fine window. Three parts of Algorithm 12 reach it. The first is the far part search of Section 10.3, which runs on every file. If the global offset settles on one segment, the slices that cover another segment lie at that segment's offset, and two of them within η of each other, each more than 45 s from the global offset and each with $\Lambda_j \geq 6$, make a second part. The eight slices are checked first and then sixteen slices aligned within ±90 minutes. A file with a second part goes to the recut path, whose grid reaches 600 s each way in steps of 100 ms and gains an island around every backed far offset (Section 9.1), so every segment lies inside the search. The second is the wider split of Section 10.5. When the slices agree well enough for the shifted path and no second part turns up, the fine split runs within ±45 s, and a piece that belongs further out ends up pressed against the edge of the window. A piece within a tenth of the window of its edge, or pieces that the slices contradict in the sense of Definition 10.3, send the file through the recut search with the same anchor, and that answer replaces the fine one when it is worth splitting with a reach of 600 s and leaves the file with a higher lock once the pieces are moved. The third is the verdict of Section 10.6. In 2.0.0 the split ran only after a *solid* verdict on the global offset, and a cut keeps any one offset from standing out, so a cut file could come back *unsure* and never be split. The fine split now runs for an *unsure* answer too, and an answer in pieces is judged by Λ_{after} on the finished file, which is high when the pieces are right, whatever the global offset scored.

The five films that 2.5.0 still does not pass in auto are a mixed group. The three *Godfather* films have spreads of 57, 59.5 and 71.5 s. Two of them fail the normal test as well, and the third is *unsure* in every run, so the problem there lies with the alignment of the film as a whole. *Eraserhead* (30 s, two cuts) is partial in both modes, two of three segments. *Manchester by the Sea* has a single cut with a spread of

only 23.5 s, fails in auto with *nothing* and passes when forced with *solid*. If, as the verdict suggests, its global offset had a lock below 6, the shifted path never tried the split, since a *nothing* answer is not split, and forcing the split skips that gate. This is read from the verdict. Whether the gate should be lowered is an open question, because the same gate is what keeps a subtitle for a different film from being cut into pieces that each find something.

12.11 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). 2.5.0 had a lower error than 2.0.0 on 11 of the twelve films, landed within 100 ms on 5, as 2.0.0 did, and alass on 4. On four films the gain was between 26 and 29 ms, and on *Requiem for a Dream* it was 345 ms. ffsubsync reported an 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 every tool, 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. *Stalker* is the odd one: 2.5.0 passes it with an error of 101 s against the download, close to alass's, while 2.0.0 failed it with 89 s.

12.12 A subtitle for the wrong film

A subtitle for a documentary about Oppenheimer was run against the film *Oppenheimer*. The right answer is to refuse. LAPSE 2.5.0 took 61.5 s, reported $\Lambda = 1.87$ and the verdict *nothing*, and left the original alone, with it's guess in a sidecar; 2.0.0 had done the same in 83.3 s with $\Lambda = 2.14$. 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.[13]

12.13 Failures film by film

The Godfather and *The Godfather Part II* failed the normal test for every tool; 2.5.0 marked both *unsure* and kept the originals. *The Godfather* passes when the split is forced, still *unsure*, and Part II fails every split run of every tool. *The Godfather Part III* passes the normal test only with LAPSE, is partial in auto and passes when forced, *unsure* throughout, so its correct answers went to a sidecar. *Eraserhead* has two cuts and gets two of three segments in both of 2.5.0's split runs, *unsure* in both; alass gets the same partial result. That leaves *Manchester by the Sea*, discussed above, which with *Ex Machina* and *Zoolander* is one of three films that ffsubsync's split mode passes and alass fails.

12.14 What the benchmark does not show

The benchmark compares tools and versions. 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 6.5), the comparison Figure 3 makes on one film;
- 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;

[13] ffsubsync can skip writing with `--skip-sync-on-low-quality`. It is off by default, and the benchmark runs every tool as shipped.

- a fixed penalty of 7 in place of $\max(3, \ln N)$;
- no lead-in, no far part search, and the 2.0.0 verdict before the split.

The evidence for individual design choices is therefore what this report collects along the way: the development numbers quoted with each change, the one film of Figure 3, and the worked example.

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`		first benchmark published
19 to 22 August	`69b3258` …		eight more subtitle formats; bitmap tracks as reference
9 September	`ee7dadf`		snapping to picture cuts
21 September	`8eb820b`		batch mode
24 September	`c76f550`	2.2.3	version described in the previous edition
2 October	`e540467`	2.2.4	*nothing* below 6; 16 bit back pointers; joined cut by running totals
5 to 20 October	`5e00e6a` …	2.3.0	credit lines dropped; `--from`; `--whisper`; ratios between the pairs; chapters; far FFT plan; irregular split grid with a free jump; baseline floor
25 October	`3b86d31`	2.3.1	overlapping cues glued in split mode; partial caches listened to again
1 to 12 November		2.5.0	lead-in; verdict on the finished file; framerate chosen after the split; far part search on every file; three minute pieces; `--threads`
13 November	`9ef92ea`		benchmark of 2.5.0 published

Table 8. Timeline of the changes that affect synchronisation. Commits that only touch packaging, the web interface or documentation are left out, and so are 2.3.2 and 2.3.3, which fixed formats and builds.

The first five weeks produced most of the ideas and most of the mistakes, and the mistakes were silent. The integer accumulators returned an offset of zero with no error. The decimating resampler produced a timeline 10% 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. Each time the program ran to completion and printed plausable numbers, and the failure only showed up when I checked a result against a film where I knew the right answer. 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 verdict came out of the same experience.

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 directly whether an offset lines the file up better than chance would. Propositions 6.4 and 6.5 are the after the fact account of why the second step worked as well as it did.

Speed changed what was affordable. The first search took minutes; after 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. In 2.5.0 the same budget pays for sixteen more slices over ±90 minutes on every file, and for splitting a drifting file under each of two neighbouring framerates before choosing one.

The October and November work had a different shape from the summer's. The failures it found were mostly of the most expensive kind, wrong answers called *solid*. A file with half its cues 20 minutes away came out shifted and solid; a part pressed against the edge of the fine window came out solid and 8 s out; a drifting file with a cut came out with the neighbouring framerate. I found them on long synthetic films or on a development set of 20 episodes and films with twelve kinds of damage, and checked every fix against the same set before it went in.

Some choices were made against features. The engine does not match words against the subtitle, which would give exact anchors and tie it to a language; with `--whisper` a recogniser is allowed only to say where words are. It gives the current timing of a file no advantage beyond including offset zero among its candidates and leaving the file alone when nothing would move, 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 14.4. 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.

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 (37), τ_B, the thresholds 6 and 8, and the 400 ms agreement window were all adjusted while looking at these 39 films and the failures on them. After the first benchmark the tuning moved to other material, synthetic long films and a development set of 20 episodes and films that is not published, but the targets of that work were chosen from what the first benchmark had shown. The weak point of 2.0.0 was automatic splitting on these films, and the largest gain of 2.5.0 is automatic splitting on the same films. Some of that gain will carry over to films the engine has not seen and some will not, by an amount nobody knows. alass and ffsubsync were not tuned on these films at all. Only a set of films held out from development could settle that, and there is none.

I designed, ran and scored the benchmark myself, and I also wrote one of the tools in it. The pass mark is my own judgement on watching each film, around a soft limit of 250 ms. A second person scoring the same outputs blind would make the counts more credible.

The other tools. alass and ffsubsync keep their August results. Their versions have not changed, and the machine is the same, so the outcomes should be the same; the times are three months older than those of 2.5.0, and single runs at that. Differences of a few per cent in time between the two rounds mean nothing. Both ran with their defaults, which is what a user gets, and either might do better with other settings. ffsubsync's split mode ran with its own default penalty of 5.

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 were 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. Much of the later development was done on television episodes, which the benchmark does not contain.

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. The free jump of Section 9.3 was written for the harder case, and the benchmark cannot show whether it helps.

Features outside the benchmark. `--from`, `--whisper`, chapter hints and the continuous ratio search have no benchmark numbers. What this report says about them comes from development material.

14.3 Construct and statistical validity

A pass means a viewer watching the film would not notice an offset. The 250 ms behind it is soft, so a pass is no guarantee that every cue is within it, and the millisecond error, the one continuous measure, is taken against the original download, which is itself sometimes out of sync (Section 12.11). 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 7. The pass mark also ignores how many cuts a tool made, which Section 12.7 reports separately for twelve films.

With 39 films and paired binary outcomes, a test needs a lopsided split of discordant films to reach significance. Four of the ten comparisons in Section 12 are below 0.005 and survive a Bonferroni correction: automatic splitting in 2.5.0 against 2.0.0, forced splitting in 2.5.0 against alass and against ffsubsync, and the gap between automatic and forced splitting in 2.0.0. The lead in the normal test is consistent with LAPSE being ahead and also with chance. The verdict calibration rests on 27 films, and the interval for the precision of *solid* still reaches down to 81% in the auto column.

14.4 Limitations of the engine

A far part that one slice covers. The far part search of Section 10.3 needs two of sixteen slices to agree on the far offset. A part shorter than two slices, about an eighth of the file, can be missed. In development a four minute part 25 minutes out stayed wrong and was called *solid*. It is the one known kind of input on which the engine writes a wrong answer over the original with full confidence.

Lines out of order. The boundary step of Section 9.4 does not check the order of cues, and the worked example shows it putting four lines up to 48 s before the line in front of them. The cost is small when the moved lines are mostly sound descriptions, as there, and it is still a defect.

The gate before splitting. A file whose single offset gets *nothing* is not split in auto mode. That keeps a subtitle for another film from being cut into pieces that each find something, and it also cost *Manchester by the Sea*.

Moving a file that was right. Offset zero is always a candidate and gets no other advantage, except that a result below `--min-shift` leaves the file alone. 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 accepts the risk and relies on the backup to make a bad correction cheap to undo.

Detector warm-up. Silero starts cold eight times per three minute piece (Section 5.3), 320 times in a two hour film where 2.0.0 started 96 times. A line that starts in the first windows of a lane is detected late or not at all. This has not been measured. The benchmark shows no loss from the shorter pieces, which says only that the effect is small on these films.

Smaller things.

- The recut path keeps runs that differ by a few tens of milliseconds (Section 11.4), and reports more parts than the file has.
- The first two *Godfather* films failed the normal test for every tool and in both versions. LAPSE flags them as unsure and leaves the originals in place.
- 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 is handled for any ratio between 0.94 and 1.06. Timing that is not linear, such as a subtitle made from a variable framerate source, is not handled at all.

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 and judges the answer it is about to write. On a benchmark of 39 hard films, version 2.5.0 passed more films than alass and ffsubsync in every test, found the cuts by itself in 34 files where 2.0.0 found 19, and marked no wrong answer as solid. The benchmark is small, though, the engine was developed against the same films, and only some of the differences are clear of chance.

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

Several threads are open. No ablations were run (Section 12.14), so the design choices rest on development numbers and one illustrative film. The lock statistic could be turned into a calibrated probability by fitting it to outcomes on a larger set, and the 27 films with verdicts are too few for that. The boundary step should keep cues in order, which is a small change, and the far part search should find a part that one slice covers, which is not. The point process model of Section 6.3 suggests a weighting derived from the data, in the spirit of Knapp and Carter [18], that could replace the fixed running means; whether it would do better on films where the detector hears music as speech is not known.

The benchmark itself should be repeated with a second person scoring the results blind. That is the cheapest item on the list, since the films and the injection scripts are still at hand, and I would like a set of films I did not develop against to go with it.

Acknowledgements

Carl Johan Bangsgaard designed the LAPSE logo, and the wordmark on the title page follows it. Thanks also to the user of the Jellyfin plugin whose report of an already correct Croatian subtitle being moved ended up in Section 14.4.

References

[1] Alan Agresti and Brent A. Coull. Approximate is better than “exact” for interval estimation of binomial proportions. *The American Statistician*, 52(2):119–126, 1998. doi: 10.1080/00031305.1998.10480550.

[2] M. S. Bartlett. The spectral analysis of point processes. *Journal of the Royal Statistical Society. Series B (Methodological)*, 25(2):264–281, 1963. doi: 10.1111/j.2517-6161.1963.tb00508.x.

[3] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[4] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965. doi: 10.1090/S0025-5718-1965-0178586-1.

[5] D. J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes. Volume I: Elementary Theory and Methods*. Probability and its Applications. Springer, New York, second edition, 2003. ISBN 0-387-95541-0. doi: 10.1007/b97277.

[6] Victor Efimov. Sushi: Automatic subtitle shifter based on audio. https://github.com/tp7/Sushi, 2014.

[7] European Broadcasting Union. Specification of the EBU subtitling data exchange format. Technical Specification Tech. 3264-E, European Broadcasting Union, Geneva, February 1991.

[8] FFmpeg Developers. FFmpeg: A complete, cross-platform solution to record, convert and stream audio and video. https://ffmpeg.org/, 2026. Accessed November 2026.

[9] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. doi: 10.1109/JPROC.2004.840301.

[10] Georgi Gerganov. whisper.cpp: Port of OpenAI's Whisper model in C/C++. https://github.com/ggml-org/whisper.cpp, 2022.

[11] Frank R. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974. doi: 10.1080/01621459.1974.10482962.

[12] Fredric J. Harris. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978. doi: 10.1109/PROC.1978.10837.

[13] Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. doi: 10.1214/aoms/1177703732.

[14] Brad Jackson, Jeffrey D. Scargle, David Barnes, Sundararajan Arabhi, Alina Alt, Peter Gioumousis, Elyus Gwin, Paungkaew Sangtrakulcharoen, Linda Tan, and Tun Tao Tsai. An algorithm for optimal partitioning of data on an interval. *IEEE Signal Processing Letters*, 12(2): 105–108, 2005. doi: 10.1109/LSP.2001.838216.

[15] Kaegi. Language-agnostic subtitle synchronization. Bachelor thesis, September 2019. Thesis: https://github.com/kaegi/alass/blob/master/documentation/thesis.pdf. Reference implementation: https://github.com/kaegi/alass.

[16] J. Kiefer. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4(3):502–506, 1953. doi: 10.1090/S0002-9939-1953-0055639-3.

[17] 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. doi: 10.1080/01621459.2012.737745.

[18] Charles H. Knapp and G. Clifford Carter. The generalized correlation method for estimation of time delay. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(4):320–327, 1976. doi: 10.1109/TASSP.1976.1162830.

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

[20] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.

[21] ONNX Runtime developers. ONNX Runtime. https://onnxruntime.ai, 2018.

[22] Daniel Pirch. libfvad: voice activity detection (VAD) library. https://github.com/dpirch/libfvad, 2018. Standalone fork of the WebRTC VAD engine.

[23] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR, 2023.

[24] J. Ramírez, J. M. Górriz, and J. C. Segura. Voice activity detection. fundamentals and speech recognition system robustness. In Michael Grimm and Kristian Kroschel, editors, *Robust Speech Recognition and Understanding*, pages 1–22. I-Tech Education and Publishing, Vienna, Austria, 2007. doi: 10.5772/4740.

[25] Peter J. Rousseeuw and Christophe Croux. Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88(424):1273–1283, 1993. doi: 10.1080/01621459.1993.10476408.

[26] 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. doi: 10.1109/TASSP.1978.1163055.

[27] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. doi: 10.1214/aos/1176344136.

[28] Otto Seiskari. autosubsync: Automatic subtitle synchronization tool. https://github.com/oseiskar/autosubsync, 2018.

[29] Pranab Kumar Sen. Estimates of the regression coefficient based on Kendall's tau. *Journal of the American Statistical Association*, 63(324):1379–1389, 1968. doi: 10.1080/01621459.1968.10480934.

[30] Silero Team. Silero VAD: pre-trained enterprise-grade voice activity detector (VAD), number detector and language classifier. https://github.com/snakers4/silero-vad, 2024.

[31] Michał Szymaniak. subsync: Subtitle speech synchronizer. https://github.com/sc0ty/subsync, 2018.

[32] Henri Theil. A rank-invariant method of linear and polynomial regression analysis, I, II, III. *Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen*, 53:386–392, 521–525, 1397–1412, 1950. Also in Indagationes Mathematicae 12, 85–91, 173–177, 467–482.

[33] Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020. doi: 10.1016/j.sigpro.2019.107299.

[34] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.

A Constants

Stage	Constant	Value
Input	minimum cues; cue length cap for scoring	10; 10 s
Input	credit and advert marks	19

Stage	Constant	Value
Input	MicroDVD rate from running times: tolerance	8%
Detection	sample rate; piece length; priming	16 kHz; 180 s; 3 s
Detection	pieces in flight	cores up to 8, or `--threads`
Detection	resampler	16 taps, 256 phases
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
Spans	lead-in, Silero only	150 ms
Spans	Whisper: margin around a span; minimum words	±300 ms; 50
FFT search	bin; close reach; far reach	40 ms; ±15 min; ±90 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_B; whole file draws; slice draws; slices	450 ms; 96; 48; 8
Lock	weights; slice clip; spread floor	0.6, 1.6; $[-1, 3]$; $1/N$
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	continuous range; narrow range	0.94 to 1.06; ±0.003
Drift	grid step; second scan; golden section	$\max(10^{-4}, 2400\text{ ms}/T)$; step/6; 12 iterations
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
Modes	second part: distance; lock; agreement; wide slices	45 s; 6; 400 ms; 16 within ±90 min
Verdict	*nothing* below; *solid* from; agreeing slices	$\min(6, \theta)$; $\theta = 8$; 3
Verdict	contradiction: lock near, far; distance; reach	4 within 45 s, 5 beyond; 2 s; ±15 min
Split	recut grid; fine grid; islands	±600 s at 100 ms; ±45 s at 25 ms; ±45 s
Split	backed slice: second slice within; or lock	1 s; 7
Split	penalty; at a chapter; free jump	$\max(3, \ln N)$; $p/2$ within 2 s; $2p$
Split	boundary reach; chapter bonus	60 cues; half a cue
Split	run kept: lock; neighbour within; backed within	7; 45 s; 1 s
Split	worth splitting: run length; reach; range	12 cues; 60 s (600 s wide); 120 ms
Split	wider search when a piece is within	10% of the window edge
Restart	backward jump; part size; part search	60 s; $\max(30, N/10)$ cues; ±90 min
Joined	slices; jump; confidence floor	16; 5 min; 0.02
Snap	window; keyframes; median gap; timer test	120 ms; 8; 400 ms; 100 ms
Snap	GOP cap; shortest line after an end snap	3 gaps within 80 ms; 500 ms
Cache	minimum spans; coverage	20; whole film

B Results per film

The tables in this appendix are generated from `docs/benchmarks.md` at commit `9ef92ea`. ✓ is a pass, ◦ a partial result with the share of segments that landed where the source gives it, and ✗ a failure.

Table 9. Normal test. Times in seconds. For LAPSE the mark is that of the cached run. [c] crashed.

	LAPSE 2.5.0			LAPSE 2.0.0		alass		ffsubsync	
Film	first	cached		first					
2001: A Space Odyssey	11.36	0.76	✓	13.41	✓	21.60	✓	2.21	✗
Andrei Rublev	4.46	0.51	✓	5.32	✓	6.84	✓	11.64	✓
Apocalypse Now	7.82	1.23	✓	9.41	✓	14.86	✓	22.13	✓
Badlands	8.97	0.56	✓	9.93	✓	14.10	✓	13.90	✓
Barry Lyndon	13.09	1.10	✓	16.14	✓	30.19	✓	27.83	✓
Blue Velvet	5.28	0.55	✓	6.74	✓	8.37	✓	13.94	✓
Brazil	7.94	0.96	✓	8.86	✓	12.48	✓	18.84	✓
Challengers	10.32	0.71	✓	13.64	✓	19.78	✓	2.77	✓
Come and See	13.21	0.76	✓	15.94	✓	0.09 [c]	✗	4.03	✗
Dancer in the Dark	2.68	0.56	✓	3.52	✓	4.47	✓	7.21	✓
Drive	4.80	0.43	✓	6.68	✓	6.68	✓	11.08	✓
Enter the Void	7.20	0.92	✓	8.89	✓	11.29	✓	18.07	✓
Eraserhead	8.82	0.43	✓	10.67	✓	5.75	✗	10.96	✓
Ex Machina	4.73	0.57	✓	5.49	✓	7.63	✓	12.26	✓
Eyes Wide Shut	7.01	0.86	✓	7.65	✓	11.34	✓	17.25	✓
Fallen Angels	7.29	0.63	✓	10.32	✓	14.60	✓	15.19	✓
Fargo	3.87	0.67	✓	4.93	✓	7.33	✓	11.14	✓
Hereditary	10.75	0.56	✓	13.25	✓	18.44	✓	17.51	✓
Inland Empire	16.82	0.91	✓	18.90	✓	32.08	✓	41.62	✓
Manchester by the Sea	5.84	1.30	✓	7.91	✓	8.65	✗	15.73	✓
Mulholland Drive	16.65	0.93	✓	20.00	✓	23.53	✓	23.20	✓
No Country for Old Men	8.49	0.62	✓	11.38	✓	17.53	✓	17.81	✓
Possession	6.50	0.49	✓	8.34	✓	5.95	✓	11.78	✓
Requiem for a Dream	10.74	0.62	✓	13.72	✓	8.79	✗	17.89	✓
Saving Private Ryan	6.46	1.38	✓	8.57	✓	12.46	✓	18.39	✓
Seven Samurai	15.39	1.05	✓	37.53	✓	11.04	✓	7.05	✓
Stalker	9.05	1.24	✓	10.82	✗	11.17	✓	18.32	✗
Synecdoche, New York	10.85	0.94	✓	13.07	✓	21.14	✓	18.40	✓
The Deer Hunter	9.23	1.22	✓	11.06	✓	13.62	✓	21.08	✓
The Godfather	12.85	2.46	✗	15.00	✗	12.46	✗	25.13	✗
The Godfather Part II	12.38	2.56	✗	17.30	✗	14.53	✗	28.75	✗
The Godfather Part III	10.78	1.40	✓	13.82	✓	11.83	✗	23.58	✗
The Hunt	7.81	0.52	✓	11.49	✓	7.42	✓	20.35	✓
The Passenger	14.98	0.56	✓	23.77	✓	9.69	✓	17.12	✓
The Seventh Seal	4.86	0.70	✓	5.82	✓	6.95	✓	11.49	✓
There Will Be Blood	27.09	0.69	✓	57.63	✓	31.55	✓	33.57	✓
Uncut Gems	15.17	1.29	✓	18.69	✓	23.51	✓	21.25	✓
Whiplash	4.25	0.54	✓	5.21	✓	7.58	✓	12.31	✓
Zoolander	7.48	0.66	✓	10.33	✓	6.82	✗	13.49	✓

Table 10. Split test. LAPSE auto had to find the cuts by itself; forced is `split` mode. ffsubsync is shown without and with `--split-penalty`.

		LAPSE 2.5.0		LAPSE 2.0.0			ffsubsync	
Film	Cuts	auto	forced	auto	forced	alass	plain	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	✓	✓	✓	✓	✓	✗	✓

		LAPSE 2.5.0		LAPSE 2.0.0			ffsubsync	
Film	Cuts	auto	forced	auto	forced	alass	plain	split
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	✗	◦ 2/3	◦ 2/3	✗	✗
Ex Machina	2	✓	✓	✗	✓	✗	◦ 2/3	✓
Eyes Wide Shut	1	✓	✓	✓	✓	✓	✗	✓
Fallen Angels	1	✓	✓	✗	✓	✗	✗	✗
Fargo	1	✓	✓	✓	✓	✓	✗	✓
Hereditary	1	✓	✓	✓	✓	✓	✗	✓
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	◦ 2/3	✓	✗	✓	✗	✗	✗
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. The twelve films of the first round were cut once.

Film	Cuts	Cut at	Offset per segment (ms)
Apocalypse Now	2	00:59:23	+33000, −21000, +33000
Badlands	3	00:25:07, 00:40:52	−21000, +45000, +12000, −21000
Barry Lyndon	3	00:44:48, 01:22:52, 02:13:49	−33000, +21000, −33000, +12000
Blue Velvet	1	00:55:20	+33000, −38500
Brazil	2	00:38:54, 01:41:59	+38500, −33000, +21000
Drive	2	00:29:49, 01:09:21	+21000, −33000, +21000
Enter the Void	3	00:31:50, 01:23:52, 01:45:36	−12000, −45000, −15000, +33000
Eraserhead	2	00:26:30, 00:58:41	−15000, +15000, −12000
Ex Machina	2	00:29:25, 01:02:30	+21000, +38500, −12000
Eyes Wide Shut	1	01:10:21	+12000, +33000
Fallen Angels	1	00:43:05	−45000, +45000
Fargo	1	00:44:28	−45000, −15000
Hereditary	1	00:56:00	+15000, −21000
Inland Empire	3	00:51:53, 01:31:06, 02:11:54	−12000, +12000, −33000, +12000
Manchester by the Sea	1	00:58:58	+15000, +38500
Mulholland Drive	1	01:14:58	+38500, −12000

Film	Cuts	Cut at	Offset per segment (ms)
No Country for Old Men	1	01:02:25	+33000, −15000
Possession	1	00:53:52	−33000, +33000
Synecdoche, New York	3	00:38:13, 01:04:41, 01:32:22	−38500, −15000, +21000, +38500
The Deer Hunter	2	00:57:45, 01:51:49	−12000, +21000, +45000
The Godfather	1	01:32:14	−45000, +12000
The Godfather Part II	1	01:31:22	+38500, −21000
The Godfather Part III	2	00:56:24, 01:43:55	−15000, −33000, +38500
The Hunt	4	00:22:03, 00:41:17, 01:11:23, 01:27:20	−15000, +15000, +38500, −15000, −33000
The Passenger	1	01:03:37	+33000, −38500
There Will Be Blood	1	00:36:16	−45000, +12000
Zoolander	2	00:26:30, 00:59:43	+12000, +38500, −45000

Table 12. Verdicts for the 27 films of the second round, each with the outcome of the same run.

	LAPSE 2.5.0			LAPSE 2.0.0		
Film	normal	split, auto	split, forced	normal	split, auto	split, forced
Apocalypse Now	*solid* ✓	*solid* ✓	*unsure* ✓	*solid* ✓	*unsure* ◦	*unsure* ✓
Badlands	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ◦	*solid* ✓
Barry Lyndon	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓
Blue Velvet	*solid* ✓	*nothing* ✓	*solid* ✓	*solid* ✓	*nothing* ✓	*solid* ✓
Brazil	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*nothing* ◦	*solid* ✓
Drive	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ◦	*solid* ✓
Enter the Void	*solid* ✓	*unsure* ✓	*unsure* ✓	*solid* ✓	*solid* ✗	*unsure* ✗
Eraserhead	*solid* ✓	*unsure* ◦	*unsure* ◦	*solid* ✓	*solid* ✗	*unsure* ◦
Ex Machina	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✗	*solid* ✓
Eyes Wide Shut	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓
Fallen Angels	*unsure* ✓	*unsure* ✓	*unsure* ✓	*unsure* ✓	*unsure* ✗	*unsure* ✓
Fargo	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓
Hereditary	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓
Inland Empire	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓
Manchester by the Sea	*nothing* ✓	*nothing* ✗	*solid* ✓	*nothing* ✓	*nothing* ✗	*solid* ✗
Mulholland Drive	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓
No Country for Old Men	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓
Possession	*solid* ✓	*solid* ✓	*unsure* ✓	*solid* ✓	*solid* ✗	*unsure* ✓
Synecdoche, New York	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✗	*solid* ✓
The Deer Hunter	*solid* ✓	*unsure* ✓	*solid* ✓	*solid* ✓	*solid* ◦	*unsure* ◦
The Godfather	*unsure* ✗	*unsure* ✗	*unsure* ✓	*unsure* ✗	*unsure* ✗	*unsure* ✓
The Godfather Part II	*unsure* ✗	*unsure* ✗	*unsure* ✗	*unsure* ✗	*unsure* ✗	*unsure* ✗
The Godfather Part III	*unsure* ✓	*unsure* ◦	*unsure* ✓	*unsure* ✓	*unsure* ✗	*unsure* ✓
The Hunt	*solid* ✓	*unsure* ✓	*unsure* ✓	*solid* ✓	*nothing* ✗	*unsure* ◦
The Passenger	*solid* ✓	*unsure* ✓	*solid* ✓	*solid* ✓	*unsure* ✓	*solid* ✓
There Will Be Blood	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✓	*solid* ✗	*solid* ✓
Zoolander	*solid* ✓	*solid* ✓	*unsure* ✓	*solid* ✓	*nothing* ✗	*unsure* ✓

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

Film	Injected	LAPSE 2.5.0	LAPSE 2.0.0	alass	ffsubsync
2001: A Space Odyssey	shift +97 000 ms	17 ✓	19 ✓	50 ✓	34 769 ✗
Andrei Rublev	shift +15 000 ms	23 ✓	35 ✓	168 ✓	0 ✓
Challengers	stretch ×1.00100	41 ✓	67 ✓	46 ✓	160 ✓
Come and See	shift +97 000 ms	24 071 ✓	24 082 ✓	*crash* ✗	33 110 ✗
Dancer in the Dark	stretch ×0.99900	384 022 ✓	384 035 ✓	304 591 ✓	195 928 ✓

Film	Injected	LAPSE 2.5.0	LAPSE 2.0.0	alass	ffsubsync
Requiem for a Dream	shift −33 000 ms	1 768 ✓	2 113 ✓	140 672 ✗	1 790 ✓
Saving Private Ryan	stretch ×1.00100	3 487 ✓	3 646 ✓	3 574 ✓	3 525 ✓
Seven Samurai	stretch ×1.04167	33 ✓	60 ✓	38 ✓	0 ✓
Stalker	shift +52 500 ms	101 243 ✓	89 302 ✗	101 286 ✓	0 ✗
The Seventh Seal	shift +33 000 ms	31 ✓	58 ✓	26 ✓	0 ✓
Uncut Gems	shift −33 000 ms	16 047 ✓	16 076 ✓	16 105 ✓	16 054 ✓
Whiplash	shift +38 500 ms	167 ✓	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
./build/lapse new.mkv subtitles.srt --from old.mkv # move to another release
./build/lapse --shift -1250 --ratio 1.0427 subtitles.srt
```

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. The report is built with `tectonic lapse.tex` after `python3 tools/make_tables.py`, which reads `data/benchmarks-2.5.0.md`, a copy of the benchmark file, and writes every number of Section 12 into `generated/`.