План: обрезка видео

> **Про интерфейс в этом документе.** Точки реза здесь находят в плеере mpv, который открывается кнопкой из раздела, и переносят в поля кнопками — либо вводят таймкодом с клавиатуры. Форма волны и прослушивание кадра, о которых говорится ниже, относятся к панели, отложенной до отдельного этапа: движок и измерения от неё не зависят и остаются в силе.

Состояние на 25 августа 2026: три паттерна из четырёх работают, превью и два ползунка стоят в интерфейсе. Не сделано только вырезание куска из середины со склейкой и кроссфейдом — оно и задумывалось отдельным этапом.

Что	Где живёт	Готово
Арифметика реза, таймкод, ключевые кадры	`system_core/core/trim_contract.py`	да
Операция для GUI	`system_core/services/media_service.py`, `trim_media`	да
Раздел «Обрезка»	`config/tool_manifest.yaml`, группа `trim`	да
Превью: кадр, волны, два фейдера	`system_core/ui_nicegui/app.py`, поля `trim_points`, `trim_file_pick`	да
CLI	`Scripts/ff-trim-start.cmd`, `-end`, `-both`, `kind: trim`	да
Проверки арифметики	`tests/test_trim_contract.py`, 57 тестов	да
Матрица форматов и методы проверки	TRIM_MATRIX_RU.md	да
Приёмка на матрице источников	ProRes, HEVC long-GOP, all-intra, 29,97 DF, VFR, 4 канала, MPEG-TS, кириллица	да

Что	Где живёт	Готово
Разделение файла надвое по точке	`pattern: split`, два выхода `_part1` / `_part2`	да
Снимок кадра в PNG/JPG	`trim_save_still`, `Transcoded/Stills`	да
Вырезание куска из середины, склейка, кроссфейд	—	нет

Ниже — исходный замысел; расхождения с ним, найденные на живом материале, отмечены прямо в тексте.

Зачем

Сегодня Media Tools конвертирует, но не режет. При этом самые частые задачи монтажа к монтажу отношения не имеют: снять заставку, отрезать хвост, убрать кусок из середины. Ради этого люди ставят комбайн, который весит гигабайты и перекодирует час ради двух склеек.

Материал — исходники с камер: MOV и MP4, H.264/H.265, ProRes, звук PCM или AAC, одна-две дорожки. Не фильмы с пятью языками, субтитрами и AC-3: там своя жизнь и свои инструменты. Из этого следует и набор параметров ниже — он про карточку с камеры, а не про мультиплексирование.

Отдельная причина — точность на дробных частотах. Разбор Shutter Encoder 20.2 (дефект держится с 19.1) показал: нарезка по таймкоду переводит время в кадры через `round(fps)`, и на 23.976 / 29.97 / 59.94 длина уезжает на множитель `1001/1000` — почти четыре секунды на часе. В Media Tools частота живёт точной дробью (`Fraction`) от `ffprobe` до команды `ffmpeg`, так что округлять негде по построению.

Что делаем

Четыре паттерна, ничего сверх:

1. отрезать начало;
2. отрезать конец;
3. отрезать с обеих сторон;
4. вырезать кусок из середины (вторым этапом — там появляется склейка).

Первые три — одна операция с двумя параметрами. Четвёртый разрезает файл надвое и склеивает, и потому решается отдельно.

Резка только по ключевым кадрам

Видео копируется, не перекодируется: `-c:v copy`. Рез ложится на ближайший ключевой кадр, файл готов за секунды, поколение не теряется.

Точная резка сознательно не делается. Она требует перекодирования всего файла, и таких инструментов и без нас достаточно. Наша сильная сторона — быстро и без потерь; смешивать эти два поведения в одной кнопке значит врать о времени работы.

Уточнение по факту работы. К ключевому кадру привязано только начало — именно оно должно попасть в точку, с которой копирование потока вообще возможно. Конец такого ограничения не имеет: он уезжает числом кадров (`-frames:v`), посчитанным по точной дроби, и ложится ровно туда, куда просили. Поэтому в интерфейсе у начальной точки написано «ближайший ключевой», а у конечной — «рез на».

Из этого следует главное требование к интерфейсу: **человеку показывается фактическая точка реза, а не запрошенная**. Ближайший ключевой кадр берётся у `ffprobe`, и рядом пишется расхождение: «ближайший ключевой: 00:01:28,4 (−1,6 с)». Тогда решение остаётся за человеком, а не оказывается сюрпризом в готовом файле.

Превью

Дёшево и достаточно. Плеера с перемоткой не делаем.

Кадр. Одна картинка из фактической точки реза, 426×240 (16:9). Такого размера хватает узнать сцену, а стоит она долей секунды:

```
ffmpeg -ss <точка> -i <файл> -frames:v 1 -s 426x240 -q:v 3 preview.jpg
```

Волна звука, ±2 секунды вокруг реза. Здесь суть: волну всего файла рисуют многие, и она бесполезна — на десяти минутах это сплошная гребёнка, по которой нельзя сказать, попал рез в паузу или в середину слова. Окно в четыре секунды превращает картинку в осциллограф: видно конец фразы, тишину, начало следующей.

Как сделано. Волна не картинка сбоку, а шкала: каждый фейдер несёт свою полосу прямо над собой и делит с ней диапазон, поэтому движок идёт по форме волны, а не рядом с ней. Верхняя

полоса — весь файл (грубая наводка), нижняя — те же десять секунд, что и точный фейдер. Обе рисуются тёмно-оранжевым по морскому фону, `showwavespic` с `scale=sqrt`: на линейной шкале обычная речь даёт волосок в три процента заливки, на корневой — читаемую форму.

Полоса всего файла стоит дорого только один раз: час звука — около двух с половиной секунд, поэтому она считается в фоне и кладётся в `workspace/trim_waves`, а дальше берётся оттуда за сотые доли секунды. Окно в десять секунд не кешируется вовсе — оно рисуется за 0,14 с и каждый раз новое.

Ширина полосы — 2560 точек: этого хватает на 4K в штатном окне 1600×900, а при растягивании картинка просто интерполируется. Волна показывает пики, а не спектр, поэтому лишняя точность здесь ничего не добавит.

```
ffmpeg -ss <точка-2> -t 4 -i <файл> \
  -filter_complex "showwavespic=s=760x150:colors=#e08a63" -frames:v 1 wave.png
```

Декодируются только эти четыре секунды, поэтому картинку можно перерисовывать на каждое движение ползунка. На волне: отметка реза по центру и засечки по секундам — без них не читается масштаб.

Стык на прослушивание (для вырезания куска). Три-четыре секунды вокруг шва, склеенные так, как будет в результате, отдаются тегом `<audio>`. Только так слышно, будет щелчок или нет.

Паттерны — переключателями секций

Четыре паттерна — это ряд кнопок-переключателей, а не список выбора. Нажатая кнопка открывает свою секцию, и в ней ровно те поля, которые нужны:

Кнопка	Что в секции
Начало	одна точка: докуда отрезать
Конец	одна точка: откуда отрезать
Оба края	две точки
Вырезать кусок	две точки + чекбокс кроссфейда

Так человек видит устройство операции до того, как её выберет, и не гадает, что скрывается за строкой списка. Лишние поля не показываются вовсе — не гасятся, а именно отсутствуют, чтобы не отвлекали.

Ползунки и превью общие для всех секций: меняется только число точек, которые ими задаются.

Два ползунка

Классическая беда длинного файла: на часовой записи пиксель мыши равен пяти секундам, и точку поставить нечем. Решение — два ползунка друг над другом, одного размера, разных цветов:

- **грубый** — весь файл целиком, шаг с точностью мыши;
- **точный** — окно в десять секунд вокруг выбранной точки, развёрнутое на всю ширину; там пиксель уже равен кадру.

Двигаешь грубый — точный перецентровывается вокруг новой точки. Двигаешь точный — уточняешь позицию внутри окна. Оба показывают одну величину, поэтому и размер у них один: это не «главный и вспомогательный», а один и тот же выбор в двух масштабах.

Что видно с первого взгляда

Форма отвечает на один вопрос: **что останется**. Поэтому режимы названы по результату — «Убрать начало», «Убрать конец», «Оставить середину», «Разделить надвое», — а на обеих полосах волны сохраняемый кусок подсвечен зелёным, и он же обведён; всё, что уйдёт, закрыто тёмной вуалью. Сравнивать числа не нужно, видно глазами.

Точки называются **IN** и **OUT** — терминами монтажа, а не «начало/конец», которые совпадали со словами в названиях режимов и путали. В режиме разделения точка одна и называется **CUT**: там ничего не выбрасывается, файл просто режется надвое (`_part1` и `_part2`), у второй части таймкод сдвинут.

Транспортная панель

Под фейдерами — ряд иконок, как в вьюере любой монтажки: три группы, разделённые чертой, и справа отсчёт «кадр из общего · время».

Группа	Что в ней
Точки	IN / OUT (или CUT) — какую точку двигают фейдеры
Перемещение	в начало, к предыдущему ключевому, кадр назад, поле времени, кадр вперёд, к следующему ключевому, в конец
Снимок	сохранить текущий кадр в PNG или в JPG, полным разрешением, в `Transcoded/Stills`

Подписей на кнопках нет — только иконки и тултипы. Единственный текст в ряду — поле времени в формате `чч:мм:сс,мс` (не таймкод и не сэмплы; это сказано в тултипе). Отсчёт «кадр из общего · время» и сводка результата идут **отдельной строкой под кнопками**: в одной строке с ними они на узком окне выдавливали кнопки за край. Ряд переносится по группам, на 920 px всё умещается без горизонтальной прокрутки.

В режиме «Оставить середину» у грубой полосы две ручки — левая ставит IN, правая OUT, а участок между ними и есть то, что останется: он подсвечен и на рейке, и на волне над ней. Точная полоса остаётся одиночной (в её десятисекундном окне второй край обычно не помещается) и следует за той ручкой, которую двигали последней.

Пульт входных данных

Файл выбирается не списком, а перелистыванием: бейдж с именем текущего файла и стрелки по краям, по алфавиту и по кругу. Бейдж занимает 60% ширины (камерные имена длинные), стоит по центру, фон у него терминальный, шрифт Cascadia Mono 11 pt полужирный — это пульт, а не поле формы. Рядом счётчик «3/12», закладка на отмеченных файлах и кнопка сведений: таблица ffprobe (кодек, точная дробь частоты, VFR, таймкод, начало таймлайна) — то, что нужно знать до реза.

Точки помнятся по файлам. У каждого файла свои A и B; шагнул к следующему — твои точки остались, вернулся — они на месте. Ключ кэша не имя, а дешёвый хеш содержимого (размер плюс первый и последний мегабайт), поэтому переименование дубля отметку не теряет. Кэш живёт в окне: закрыл программу — кэш чистый.

Запуск режет всё отмеченное разом, каждый файл по своим точкам. Резать всю папку по одним и тем же точкам инструмент больше не предлагает: у дублей не совпадают ни ключевые кадры, ни границы. Пакета «одинаково всем» больше нет нигде: CLI тоже режет один файл — тот, на который указали `AUDION_TRIM_FILE`, либо первый в Source, и обёртка говорит, какой из двух вариантов выбрала.

Как это выглядит в сборке

Комплект контролов один на всю операцию, а точек может быть две — поэтому они переключаются кнопками **A** (начало куска) и **B** (конец). Второй комплект ползунков не нужен: точка меняется, ползунки те же. Рядом всегда видна сводка «A · B · длина куска», так что переключение не прячет вторую точку.

Раскладка блока: слева «телевизор» — кадр и под ним кнопки перехода, справа две подгруппы со своими заголовками, «Весь файл» и «Окно 10 секунд», в каждой полоса волны и фейдер под

ней. Ряд управления — кнопки A/B, покадровые стрелки, поле времени и сводка «A · B · длина куска» — идёт снизу, под всем блоком. Превью стоит сбоку, а не под ползунками, чтобы форма не съедала высоту окна.

Ползунки сделаны фейдерами: рейка 26 px тёмно-морского цвета с обводкой, оранжевый движок 14×34. Высоту рейки Quasar пишет инлайном из `track-size`, так что она задаётся пропом, а не стилем.

Покадровая точность — тремя способами: стрелки ◀ ▶ рядом с маркером, шаг ровно кадр; горизонтальный скролл (MX Master и подобные) — кадр за щелчок на точном ползунке и полсекунды на грубом; и просто ввод времени в поле.

Точки A и B стоят на обеих полосах зелёным пунктиром, а текущее положение фейдера — светлой сплошной чертой. Поэтому кусок, который останется, виден целиком, даже когда двигаешь только один его край.

Под кадром — навигация, которая относится именно к резке: в начало файла, к предыдущему ключевому кадру, к следующему, в конец. Прыжок по ключевым кадрам здесь не украшение: начало куска всё равно ляжет на один из них.

Кадр перерисовывается через 0,35 с после того, как рука остановилась: сама отрисовка стоит ~90 мс, но дёргать её на каждый пиксель протяжки незачем.

Кроссфейд

Чекбокс, всегда на виду. Включён — стык звука сводится за 20–30 мс, щелчка нет.

Важно: кроссфейд означает пересборку звука, скопировать дорожку уже нельзя. Но видео остаётся `-c:v copy`, а перекодируется только аудио — секунды на файл. То есть галочка не превращает быструю резку в медленную, и обещание «быстро» остаётся честным.

Подсказка, которую стоит сделать: если по обе стороны шва сигнал не в нуле — подсветить чекбокс. Человек не обязан знать, когда бывает щелчок.

Параметры

Критерий отбора один: параметр либо ничего не стоит при копировании потоков, либо ломает обещание «две секунды вместо двадцати минут». Дорогого здесь нет.

Таймкод — первое, что нельзя потерять

Камера пишет таймкод в файл, и по нему на монтаже синхронизируют камеры между собой и со звуком с рекордера. Обрезка обязана сдвинуть его правильно: начало нового файла — это старый TC плюс отрезанное.

Отдельная беда, которую надо проверять и говорить вслух: при перекладывании MOV → MP4 дорожка `tmcd` часто теряется молча. Материал после этого выглядит целым, а синхронизировать его уже нечем. Если таймкод не переносится — сказать об этом до запуска, а не оставить человека выяснять это на монтаже.

По умолчанию TC сдвигается. Обнуление — отдельным выбором, для случаев, когда файл уходит наружу как самостоятельный.

Что показала проверка. FFmpeg 9.0.1 переносит дорожку `tmcd` при копировании MOV → MP4, но значение оставляет прежним — сдвиг всё равно надо писать самому через `-timecode`. В MKV таймкод живёт текстовым тегом, а не дорожкой, и монтажка может его не прочитать: об этом сказано предупреждением до запуска. Ещё одно предупреждение — PCM в MP4 (нестандартный тег `ipcm`), для камерного звука дом — MOV.

Частота — та, что в файле

`ffprobe` отдаёт точную дробь, поэтому ничего не домысливается: снятое на 23,976 объявляет `24000/1001`, снятое на честные 24,000 — `24/1`. Оба читаются как написано.

Это отменяет прежнее правило «целое NTSC читаем как 1000/1001». Оно исходило из того, что контейнеры округляют, — они не округляют. ARRI Alexa Mini, снимавшая ровно 24,000 (260 кадров на 10,8333 с — измерено), объявляет `24/1`, и множитель превращал верную частоту в неверную: пять кадров за четыре минуты, ровно та же ошибка `1001/1000`, ради которой галочку и заводили, только в другую сторону.

Галочки больше нет ни на панели, ни в CLI. Частота, которую не удалось прочитать, отклоняется, а не заменяется догадкой.

Что можно менять бесплатно

Параметр	Зачем
Контейнер: как есть / MP4 / MOV / MKV	MOV с камеры → MP4 для монтажной, без единого пережатия. ProRes при этом остаётся ProRes
Звук: копировать / убрать	Немой файл для монтажа или для гифки — частый случай

Параметр	Зачем
Канал звука: оба / первый / второй / свести в моно	У камеры это обычно пушка и петличка в двух моно-каналах. Выбор канала — работа внутри дорожки, при PCM бесплатная
`faststart`	Только когда на выходе MP4: заголовок в начало, файл начинает играть не скачавшись целиком

Метаданные камеры — беречь

Модель, режим съёмки, гамма, имя ролика. На монтаже это контекст, без которого материал обезличивается. Чистка метаданных — операция для публикации в сеть, и живёт она в другом разделе; здесь всё переносится как есть.

Чего здесь нет

Битрейта, кодека, разрешения, частоты кадров, нормализации громкости. Как только они появляются в окне обрезки, их трогают — и получают двадцать минут вместо двух секунд, не поняв, за что заплатили.

Нормализация вредна и по существу: у исходников она убивает запас и связь между дублями. Для записи совещания она, наоборот, нужна — но это отдельный профиль («запись встречи → нормализовать и пожать»), а не галочка в резке.

Выбор языковых дорожек и работа с AC-3 — тоже не сюда. Это мультиплексирование фильмов, для него есть свои инструменты.

Как ложится в проект

Операции описаны строкой в `config/script_profiles.yaml` и разведены по `kind` в диспетчере `Scripts/Common/script_runner.py`. Обрезка — новый `kind: trim` рядом с `encode`, `editing`, `remux`, `fps`. Чтение источника, вывод, логи и отчёт уже работают и переиспользуются.

Оценка: первые три паттерна с превью — день-полтора. Вырезание куска со склейкой и кроссфейдом — ещё день, отдельно.

Что источник может сделать с резом

Проверка на разношёрстном материале дала пять правок, которые стоит помнить:

- **Кадр у самой границы конца принадлежит куску.** Конец округляется вверх: на 29,97 тридцатая секунда — это кадр 899,1, а кадр 899 стоит на 29,9966, то есть внутри.
- **`-frames:v` точен, только когда `-ss` попал ровно на ключевой кадр.** Иначе ffmpeg считает кадры от предыдущего ключевого и выбрасывает те, что до запрошенной точки, — кусок выходит короче ровно на этот зазор.
- **HEVC с B-пирамидой отдаёт на 2–4 кадра меньше, чем просили.** Недобор устойчив для потока, но не выводится из `has_b_frames`, поэтому он измеряется: кадры пересчитываются, и при недоборе рез повторяется один раз с поправкой.
- **Переменная частота запрещает кадровую арифметику.** `avg_frame_rate` — среднее, и счёт по нему покрывает не тот отрезок (двенадцать секунд просили, шестнадцать получили). Такой источник режется по времени, о чём сказано в логе.
- **Таймстемпы не всегда начинаются с нуля.** MPEG-TS обычно стартует с 1,44 с; точки человека считаются от первой картинки, ffmpeg — от начала таймлайна потока, поэтому смещение снимается с ключевых кадров и возвращается при seek.

Отдельно про MPEG-TS: индекса в нём нет, seek интерполируется по байтам — в середине файла точно, у самого конца может промахнуться. Об этом предупреждение, а слишком короткий результат считается неудачей, а не успехом со странной длительностью.

Синхрон звука: проверено, а не обещано

Рез, который двигает одну дорожку и не двигает другую, вылезает позже всего — на монтаже, а не в логе. Поэтому синхрон меряется меткой, которая есть в обеих дорожках одновременно: белый кадр в начале каждой секунды и щелчок 20 мс в тот же момент. Видео читается через `signalstats` с `fps_mode passthrough` (иначе фильтр дублирует кадры под свои часы), звук — по сэмплам; каждая вспышка сопоставляется со своим щелчком по фактическим меткам времени файла.

Отклонение **от исходника** после реза:

Контейнер	Изменение
MP4	0…2 мс
MOV	0…5 мс

Контейнер	Изменение
MKV	1…3 мс
MXF	0…5 мс (после правки ниже)

Проверены все режимы — убрать начало, убрать конец, оставить середину, разделить надвое — на AAC, PCM 24 бит и ProRes, включая выбор канала (там звук пересобирается).

MXF отдельно. Сначала он сдвигал звук на 9–18 мс, и это было неприемлемо: MXF — рабочий формат Premiere. Замер по сэмплам показал причину: при копировании PCM в MXF в файл попадают 384 сэмпла между границей чанка и точкой реза — 8 мс звука, который принадлежит моменту до реза. Если тот же PCM переписать той же разрядности, кодировщик режет ровно по запрошенному сэмплу, и звук совпадает с эталонной нарезкой байт в байт. Так и сделано: MXF теперь меряется наравне с MOV, а в логе сказано, что перезапись была (PCM→PCM поколение не тратит).

Сжатый звук в MXF не попадает вовсе: мультиплексор отказывает и AAC, и AC-3, и MP3, и FLAC, и ALAC. Такой файл пропускается до запуска с объяснением и подсказкой (MOV возьмёт как есть), а не падает с кодом 4294967274.

Проверка на настоящем материале

Canon `MVI_*.MP4`: HEVC Rext 4:2:2 10 бит UHD, 23,976, звук LPCM `pcm_s16be` (`twos`), таймкод 15:58:57:17. Прогнаны все режимы — кадры точны везде, разделение дало 144 + 170 = все 314 кадров исходника, таймкод сдвинулся верно, а звук нашёлся в исходнике **дословно**: кусок начинается на 48 сэмплов (одна миллисекунда, сороковая доля кадра) после точки реза, то есть скопирован, а не пережат.

Нашлись две вещи, которых синтетика показать не могла:

- **Терялись камерные метаданные.** `make`, `model` и бренд `mp42hvc1CAEP` лежат в боксе mdta, и `-map_metadata 0` их не переносит — мультиплексор писал общий `isomiso2mp41`. Добавлен `-movflags use_metadata_tags`, после чего теги сверены с исходником по одному.
- **Выбор канала менял формат звука.** `pcm_s16be` превращался в `pcm_s16le` — те же сэмплы, переставленные байты, и другой тег в файле. Теперь порядок байт берётся из источника.

Что осталось на второй этап

Вырезание куска из середины: файл режется надвое, части склеиваются, на шве включается кроссфейд 20–30 мс. Кнопка «Вырезать кусок» появится в ряду паттернов вместе с реализацией — сейчас её нет, чтобы не обещать несделанного.

Чего не делаем

- Таймлайна с дорожками. Это другой продукт.
- Точной резки с перекодированием. Смотри выше.
- Волны всего файла. Она красива и бесполезна.
- Догадок о намерениях: если рез уехал на ключевой кадр, об этом сказано, а не исправлено втихую.