

ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ/INFORMATICS AND INFORMATION PROCESSES

DOI: https://doi.org/10.60797/IRJ.2026.169.17 EDN: GDJERS

ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ СОВМЕЩЕНИЯ РАЗДЕЛЬНО ЗАПИСАННЫХ ВИДЕО- И АУДИОДОРОЖЕК ПО АРТИКУЛЯЦИИ ДИКТОРА

Научная статья

Шакирзянов М.Э.[1], Гибадуллин Р.Ф.[2,] *

[2] ORCID : 0000-0001-9359-911X;

[1, 2] Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ, Казань, Российская Федерация

* Корреспондирующий автор (landwatersun[at]mail.ru)

Предложена: 07.05.2026; Принята: 23.06.2026; Опубликована: 17.07.2026

Аннотация

Рассмотрены инженерные решения, лежащие в основе разработанного программного инструмента, который автоматически совмещает звуковую дорожку с видеорядом по движениям губ диктора. Инструмент написан на Python, имеет кроссплатформенную модульную организацию и использует ранее обученную свёрточно-темпоральную модель с контрастивной целевой функцией InfoNCE. Такая модель проецирует визуальный и акустический сигналы в единое векторное пространство. Сопоставление выполняется в два прохода. Сначала с шагом в 1 с пакетно строятся эмбеддинги для пятисекундных блоков, из которых формируется матрица косинусных сходств. Затем результат фильтруется: устраняются выбросы по медиане, контролируется монотонность аудиопозиций, а подряд идущие совпадения объединяются в общие сегменты. Пользователь может работать через окно на PyQt6 либо вызывать инструмент из терминала в пакетном режиме. Итоговый файл MP4 собирается с помощью утилиты FFmpeg. По итогам испытаний на 50 контрольных парах вида «короткий видеоролик — длинная звуковая запись» доля верно локализованных совпадений достигла 92,0 % при средней абсолютной погрешности начала корректно найденного отрезка 71 мс. Полный цикл обработки одной такой пары занимает 9,2 с при использовании видеокарты NVIDIA RTX 3060 и 37,7 с — на процессоре ноутбучного класса Intel Core i7.

Ключевые слова: программный комплекс, аудиовизуальная синхронизация, артикуляция, матрица косинусных сходств, медианная фильтрация, скользящее окно, графический интерфейс, FFmpeg, PyQt6, MediaPipe.

SOFTWARE PACKAGE FOR SYNCHRONISING SEPARATELY RECORDED VIDEO AND AUDIO TRACKS BASED ON THE SPEAKER’S ENUNCIATION

Research article

Shakirzyanov M.E.[1], Gibadullin R.F.[2,] *

[2] ORCID : 0000-0001-9359-911X;

[1, 2] Kazan National Research Technical University named after A.N. Tupolev – KAI, Kazan, Russian Federation

* Corresponding author (landwatersun[at]mail.ru)

Suggested: 07.05.2026; Accepted: 23.06.2026; Published: 17.07.2026

Abstract

The engineering solutions underlying the developed software tool, which automatically synchronises the audio track with the video sequence based on the speaker’s lip movements, are examined. The tool is written in Python, has a cross-platform modular architecture, and utilises a pre-trained convolutional-temporal model with the InfoNCE contrastive objective function. This model projects visual and acoustic signals into a single vector space. The alignment is performed in two passes. First, embeddings for five-second blocks are constructed in batches at one-second intervals, from which a cosine similarity matrix is formed. The result is then filtered: outliers are removed based on the median, the monotonality of audio positions is checked, and consecutive matches are merged into common segments. Users can work via a PyQt6 window or run the tool from the terminal in batch mode. The final MP4 file is generated using the FFmpeg utility. Based on tests conducted on 50 test pairs of the "short video clip — long audio recording" type, the proportion of correctly localised matches reached 92.0 %, with an average absolute error of 71 ms in the start time of correctly identified segments. The full processing cycle for a single such pair takes 9.2 s when using an NVIDIA RTX 3060 graphics card and 37.7 s on an Intel Core i7 laptop-class processor.

Keywords: software package, audiovisual synchronisation, enunciation, cosine similarity matrix, median filtering, sliding window, graphical interface, FFmpeg, PyQt6, MediaPipe.

Введение

Дистанция между обученной нейронной сетью и работающим инструментом синхронизации обычно оказывается значительно длиннее, чем принято считать. Сама модель способна переводить кадр области губ и фрагмент мел-спектрограммы в один и тот же эмбеддинг-вектор, однако конечному пользователю требуется не вектор, а собранный MP4-файл, в котором ранее записанная звуковая дорожка корректно совмещена с ранее снятым видео. Промежуток между этими двумя состояниями — прикладная задача со своей внутренней структурой. Она включает следующие этапы: подготовка входных потоков (приведение кадра к стандартному виду, выделение области губ, преобразование

звука в мел-представление), посегментная обработка данных через сеть, аккуратное установление соответствия между полученными векторами, удаление выбросов, восстановление непрерывной временной оси, передача данных во внешний сборщик, наглядное отображение результата в интерфейсе. У каждого из перечисленных шагов имеются собственные инженерные нюансы, которые существенно сказываются как на качестве итога, так и на эксплуатационные характеристики всей системы [1].

Подавляющее большинство публично доступных реализаций актуальных моделей аудиовизуальной синхронизации — SyncNet [2], Wav2Lip [3], AV-HuBERT [4] и им подобные — по сути остаются исследовательским кодом. Они запускаются в Jupyter-блокнотах, рассчитаны на работу с единичным отрезком, при старте требуют заранее настроенной CUDA и строго определённых версий библиотек, а также не имеют ни оконного интерфейса, ни пакетного режима. Чтобы превратить такую модель в инструмент, который действительно можно встраивать в рабочий процесс монтажёра или журналиста, необходимо создать полноценный программный комплекс с обработкой исключительных ситуаций, поддержкой множества форматов входа и выхода, кроссплатформенным графическим интерфейсом, эффективным использованием GPU и автоматическим переходом на CPU при его отсутствии.

Вторая существенная трудность связана со сменой постановки задачи. Вместо оценки одного смещения между двумя заведомо согласованными фрагментами требуется находить соответствия между коротким видеоотрывком и длинной аудиозаписью. Именно такой сценарий типичен для практики: у монтажёра обычно есть видеоклип длительностью 10–30-секунд (или несколько подобных клипов) и аудиозапись интервью либо подкаста продолжительностью в час и более. При этом нужно определить, в какой точке длинной записи находится фрагмент, содержательно соответствующий клипу. Простая модель, которая выдаёт для пары отрезков одно скалярное число, в этом случае непригодна. Необходимо построить отображение оси видеовремени в ось аудиовремени — оно должно отражать структуру всего материала.

В представленной статье излагается архитектура и реализация программного комплекса, в котором обе указанные задачи решаются совместно. Используемая нейросетевая модель — свёрточно-темпоральная сеть, обученная с контрастивной функцией потерь InfoNCE, — подробно описана и обоснована в [5]. Основной акцент здесь сделан на инженерной составляющей: модульной структуре комплекса, реализации двухступенчатой процедуры сопоставления, постобработке найденных совпадений, сборке итогового файла с помощью FFmpeg [6], а также на устройстве графического интерфейса и пакетного режима. Приводятся характеристики быстродействия и точности сопоставления, измеренные на тестовой подвыборке из 50 пар «короткое видео — длинная аудиозапись», для которых истинное совпадение известно.

Архитектура программного комплекса

2.1. Принципы построения

В качестве среды реализации выбран Python 3.10; комплекс выполнен в виде модульного кроссплатформенного приложения. Нейросетевые операции выполняет PyTorch, обработка видео опирается на связку OpenCV и MediaPipe, обработка аудио — на Librosa и soundfile, оконный интерфейс построен на PyQt6 (с интегрированным через Matplotlib просмотром графиков), а сборку итогового файла осуществляет внешний процесс FFmpeg. На этапе проектирования последовательно применялся ряд принципов, которые заранее задают общий облик кодовой базы и упрощают её последующее сопровождение [7].

Функциональная декомпозиция. Кодовая база разбита на шесть функциональных подсистем: подготовка видеопотока, подготовка аудиопотока, синхронизация, обучение и оценка модели, графический интерфейс и сбор обучающих данных. Каждой подсистеме отведён собственный Python-пакет, и каждая решает строго одну задачу. Обмен данными между подсистемами организован через простые структуры — массивы NumPy, тензоры PyTorch и словари, описывающие результат.

Централизованная конфигурация. Ключевые числовые параметры — разрешение области губ (96×96 пикселей), количество мел-фильтров (80), длительность блока грубого выравнивания (5 с), шаг скользящего окна (1 с), размер мини-пакета при инференсе (64 фрагмента), а также пороги постобработки — собраны в одном конфигурационном файле config.py. Такое решение существенно повышает воспроизводимость экспериментов и упрощает подбор настроек под конкретный сценарий применения.

Двойной режим вычислений (CPU и GPU). Все компоненты, взаимодействующие с нейронной сетью, при запуске проверяют наличие CUDA-совместимого ускорителя: при его обнаружении задействуют GPU, при отсутствии — автоматически переходят на CPU. Эта возможность принципиально важна, поскольку среди потенциальных пользователей комплекса — не только обладатели рабочих станций с дискретной видеокартой, но и журналисты или монтажёры, работающие на обычном ноутбуке без выделенного GPU.

2.2. Структура модулей и поток обработки

Файловая иерархия комплекса спроектирована по принципу «одна папка — одна подсистема». Единая точка входа — модуль av_sync/main.py; режим работы выбирается первым позиционным аргументом командной строки: gui (открытие графического окна; используется по умолчанию, если аргументы отсутствуют), preprocess <dir> (пакетная подготовка указанного каталога с видеофайлами), train (запуск цикла обучения), evaluate (расчёт метрик на отложенной выборке), download <url или поисковый запрос> (загрузка одиночного видеоматериала по ссылке либо по запросу к видеохостингу), sync <видео> <аудио> [output] (синхронизация одной пары без графического интерфейса). Массовое наполнение датасета одной команды выполняет отдельный высокоуровневый скрипт download_training_data.py, описанный в разделе 6. Универсальность точки входа делает комплекс одинаково удобным как для интерактивной работы, так и для встраивания в автоматизированные конвейеры массовой обработки мультимедиа.

Конвейер обработки в пользовательском режиме «синхронизация» устроен следующим образом. Подсистема подготовки видео принимает входной файл, в каждом кадре находит лицо с помощью MediaPipe FaceLandmarker, по 11 опорным точкам выделяет прямоугольную область губ, приводит её к фиксированному размеру 96×96 пикселей и складывает результат в трёхмерный массив формы $(T, 96, 96)$, где T — число кадров. Подсистема подготовки аудио загружает звуковую дорожку через Librosa, переводит её в моноканальный сигнал с частотой 16 кГц, рассчитывает логарифмическую мел-спектрограмму на 80 фильтрах с шагом в 10 мс и нарезает её на отрезки, согласованные с видеокадрами по длительности (по 4 столбца на кадр). Подсистема синхронизации поднимает обученную модель из контрольной точки best.pt, прогоняет видео- и аудиоданные через соответствующие энкодеры, формирует матрицу косинусных сходств и выполняет постобработку. На выходе получается список временных сегментов с указанием границ как в видеовремени, так и в аудиовремени, а также сопровождающими их уровнями уверенности; этот список передаётся в модуль-обёртку над FFmpeg, который и собирает финальный MP4-файл.

2.3. Распределение ответственности между подсистемами

Содержание задач каждой подсистемы и набор её ключевых внешних зависимостей представлены в таблице 1.

Таблица 1 - Подсистемы программного комплекса и набор их внешних зависимостей

DOI: https://doi.org/10.60797/IRJ.2026.169.17.1

Подсистема	Функция	Зависимости
Предобработка видео	Локализация лица, выделение области губ, нормализация	OpenCV, MediaPipe
Предобработка аудио	Загрузка, ресэмплирование, вычисление мел-спектрограммы, сегментация	Librosa, soundfile
Синхронизация	Пакетное вычисление эмбеддингов, матрица сходств, постобработка	PyTorch, NumPy
Обучение и оценка	Цикл обучения, валидация, расчёт retrieval-метрик	PyTorch, scikit-learn
Графический интерфейс	Главное окно, выбор файлов, прогресс, визуализация результата	PyQt6, Matplotlib
Сбор датасета	Автоматическая загрузка обучающих видеоматериалов по запросам	yt-dlp

Принятая декомпозиция позволяет независимо развивать и проверять каждую подсистему в отрыве от остальных. Так, при появлении более точной нейросетевой модели достаточно заменить файл чекпоинта, не затрагивая остальной код. Чтобы добавить поддержку нового видеоформата, требуется доработать только подсистему подготовки видео. Графический интерфейс отделён от вычислительной части и при необходимости может быть заменён на веб-фронтенд либо встроен в стороннее приложение для видеомонтажа.

Подсистема предобработки данных

3.1. Извлечение области губ из видеопотока

Поступающий видеопоток обслуживает класс $VideoProcessor$ из модуля av_sync/processing/video_processor.py. На вход подаётся произвольный видеофайл, на выходе образуется тензор формы $(T, 96, 96)$, где T — число кадров при частоте 25 Гц. Внутренняя логика модуля построена так, чтобы к моменту подачи на вход нейронной сети область губ имела одинаковый размер, ориентацию и цветовое представление вне зависимости от исходных параметров файла.

Локализацию лица обеспечивает MediaPipe FaceLandmarker [8] — облегчённая свёрточная модель, которая для каждого кадра выдаёт 478 опорных точек. Для очерчивания области губ берётся подмножество из 11 ландмарок, отвечающих за внешние и внутренние углы губ, а также за их верхние и нижние вершины. По этому подмножеству вычисляется охватывающий прямоугольник. Чтобы его координаты не «прыгали» от кадра к кадру при поворотах головы, они сглаживаются во времени экспоненциальным фильтром по соотношению $\hat{x}_t = 0,7 \cdot x_{t-1} + 0,3 \cdot x_t$ (бо́льший вес отдан предыдущей оценке — так достигается заметное сглаживание без видимого временно́го отставания). Полученный участок изображения вырезается, переводится в полутоновую (одноканальную) форму, приводится к стабильному разрешению 96×96 пикселей и записывается в выходной тензор.

Если детектор не обнаружил лица в очередном кадре (например, говорящий отвернулся или сцена содержит только крупный план без лица), используется последний валидный ограничивающий прямоугольник. Если лицо отсутствует в нескольких подряд идущих кадрах, текущий кадр заполняется нулями и помечается флагом «невалидный». При последующей синхронизации такие кадры исключаются из вычислений: их вклад в эмбеддинг блока обнуляется, а веса при усреднении соответствующим образом перенормируются.

3.2. Извлечение признаков из аудиопотока

Класс $AudioProcessor$ из модуля av_sync/processing/audio_processor.py принимает на вход звуковой файл в произвольном формате, понижает частоту дискретизации до 16 кГц, сводит сигнал в моно и строит логарифмическую мел-спектрограмму на 80 фильтрах в полосе 80 Гц – 8 кГц. Анализирующее окно — 25 мс, шаг между окнами — 10 мс; в перерасчёте на видеокадры это означает 4 столбца мел-спектрограммы на каждый кадр (при 25 Гц длительность кадра равна 40 мс). Привязка мел-спектрограммы к видеокадрам приводит к тому, что каждому видеокадру соответствует свой небольшой акустический срез размера (80,4).

Такое жёсткое временно́е соответствие между видео- и аудиоканалами — обязательное требование архитектуры сети: совместное эмбеддинг-пространство построено в предположении, что каждый видеокадр сопоставлен акустическому сегменту равной длительности. Любое нарушение этого баланса, в частности отклонение реальной частоты кадров входного видео от номинальной, ведёт к деградации синхронизации. Поэтому при неоднозначном fps подсистема подготовки видео выводит соответствующее диагностическое сообщение и приводит частоту кадров к 25 Гц линейной интерполяцией. Согласованная с ней подсистема подготовки аудио оперирует уже корректным потоком кадров.

Двухступенчатый алгоритм сопоставления

4.1. Постановка задачи и обозначения

Пусть $V = (V_1, \dots, V_{T_V})$ — последовательность кадров видео длиной T_V кадров, а $A = (A_1, \dots, A_{T_A})$ — соответствующая ей последовательность сегментов мел-спектрограммы длиной T_A, использующая тот же временно́й шаг в 40 мс. В практически интересном случае выполнено условие $T_V \ll T_A$: короткий видеоотрывок сопоставляется длинной звуковой записи. Цель алгоритма — построить набор попарно непересекающихся пар интервалов $\{(\alpha_i, \beta_i] \cdot [\gamma_i, \delta_i]\}_{i=1}^{K}$, в которых первый интервал задаёт диапазон в видеовремени, а второй — соответствующий ему диапазон в аудиовремени. При этом оба должны содержать содержательно совпадающий материал. Решение должно быть пригодно для реальных условий эксплуатации: оно должно обходиться без полного перебора всех возможных сдвигов, который заведомо неприемлем по вычислительной стоимости при типичных длительностях входных файлов.

4.2. Грубый этап: пакетное вычисление эмбеддингов

На грубом этапе и видеоряд, и спектрограмма нарезаются на перекрывающиеся блоки длительностью 5 с (125 кадров) со сдвигом в 1 с (25 кадров). Для каждого блока пакетно рассчитывается эмбеддинг соответствующей модальности с помощью обученных энкодеров E_v и E_a; результирующие векторы подвергаются L2-нормировке. Соответствующий псевдокод фрагмента алгоритма приведён ниже.

Пусть N_V и N_A — количество блоков, на которые разбиты видео- и аудиопотоки соответственно, а $\hat{V} \in \mathbb{R}^{N_V \times d}$ и $\hat{A} \in \mathbb{R}^{N_A \times d}$ — соответствующие матрицы L2-нормированных эмбеддингов размерности $d = 256$. Матрица попарных косинусных сходств между всеми блоками сводится к одной операции умножения матриц:

$$S = \hat{V} \cdot \hat{A}^{\cdot} \in \mathbb{R}^{N_V \times N_A} \tag{1}$$

В матрице каждая строка соответствует отдельному видеоблоку, а каждый столбец — отдельному аудиоблоку. Элемент S_{ij} имеет смысл косинусного сходства: значение, близкое к +1, говорит о высокой близости двух блоков в эмбеддинг-пространстве, тогда как значения вблизи нуля или отрицательные свидетельствуют об их различии. Для каждой строки выбирается наиболее похожий на неё столбец, то есть для каждого видеоблока — наиболее сходный аудиоблок:

$$j^*(i) = \underset{j \in [1, N_A]}{\operatorname{argmax}} S_{ij}, \quad c_i = S_{i, j^*(i)} \tag{2}$$

где $j^*(i)$ — индекс наиболее подходящего аудиоблока для i-го видеоблока, а c_i характеризует уровень уверенности этого совпадения. По итогам грубого этапа получается список кандидатных совпадений длиной N_V.

Выбор шага скользящего окна в 1 секунду является компромиссным. Уменьшение шага обеспечивает более плотное покрытие, но пропорционально увеличивает количество прогонов через сеть и общее время обработки. Увеличение шага, наоборот, ускоряет вычисления, однако повышает риск пропустить точную точку совпадения. На тестовой подвыборке при шаге в 1 с алгоритм корректно отыскивает совпадение в 92,0% случаев; при шаге в 2 с этот показатель падает до 88,4%, а при шаге в 0,5 с возрастает лишь до 92,8%, что не окупает почти двукратного замедления. Размер мини-пакета (64 блока) подобран таким образом, чтобы загрузить вычислительные ресурсы GPU NVIDIA RTX 3060 и при этом удерживать пиковое потребление видеопамяти в пределах 4 ГБ.

4.3. Постобработка кандидатных матчей

Грубый этап возвращает по одному кандидату на каждый видеоблок, и среди них неизбежно встречаются ошибочные — например, фрагменты с тишиной, короткими речевыми паузами или невнятной артикуляцией. Использовать такой результат без дополнительной обработки невозможно: в нём могут попадаться отдельные «выбросы», когда несколько соседних видеоблоков получили совпадения в близких аудиопозициях, а один — в семантически не связанной точке записи.

Очистка совпадений выполняется методом $\_build\_timeline$ и проходит в три этапа. На первом из них к последовательности аудиопозиций $(\gamma_i)_{i=1}^{N_V}$ применяется одномерный медианный фильтр с ядром длиной 5. Такая операция эффективно подавляет одиночные выбросы, не разрушая устойчивых трендов. На втором этапе проверяется условие монотонности: для каждой пары идущих подряд совпадений $(i-1, i)$ должно выполняться неравенство $\gamma_i \geq \gamma_{i-1}$. Содержательно это означает, что соседние видеоблоки могут указывать только на соседние или более

поздние аудиоблоки, поскольку обратный ход аудиовремени противоречит самому понятию синхронизации непрерывной речи. Совпадения, нарушающие это условие, помечаются как ошибочные и заменяются интерполированной аудиопозицией, рассчитанной по двум устойчивым соседям, либо, при невозможности интерполяции, исключаются из последующего объединения. Наконец, на третьем этапе выполняется слияние совпадений: соседние матчи объединяются в общий сегмент, если их аудиопозиции разнесены не более чем на удвоенную длительность блока (10 с). При бо́льшем разрыве открывается новый сегмент.

Каждому полученному сегменту приписывается средний уровень уверенности по всем входящим в него совпадениям; впоследствии это число используется при сортировке выдачи. Сегменты с уверенностью ниже эмпирически подобранного порога 0,3 по умолчанию сопровождаются предупреждением: модель в их корректности не уверена, и пользователю желательно проверить такой результат вручную.

4.4. Качественная оценка алгоритма

Точность работы двухступенчатого алгоритма проверялась на тестовой подвыборке из 50 пар вида «короткое видео — длинная аудиозапись». В каждой паре видеоматериал представлял собой фрагмент длительностью от 10 до 30 секунд, выделенный из записей, не пересекающихся с обучающей частью датасета. Аудиоматериал — полная запись, охватывающая этот фрагмент, с длительностью от 10 до 60 минут. Используемые метрики и их значения представлены в таблице 2.

Таблица 2 - Показатели качества двухступенчатого алгоритма на тестовой подвыборке из 50 пар

DOI: https://doi.org/10.60797/IRJ.2026.169.17.2

Метрика	Значение
Доля правильно найденных совпадений, %	92,0
Средняя абсолютная ошибка начала сегмента, мс	71
Медианная абсолютная ошибка, мс	44
Доля случаев с ошибкой начала < 100 мс, %	78,0
Доля грубых ошибок (>1 с или неверный сегмент), %	8,0

Цифры 92,0% верно отыскиваемых совпадений и 71 мс средней абсолютной погрешности по началу корректного сегмента отвечают уровню, готовому к серьёзной эксплуатации. В большинстве сюжетов видеоредактору достаточно просто принять предлагаемый результат, тогда как для 8% трудных случаев низкая уверенность модели сразу подсвечивает необходимость ручного контроля. Промежуточная категория — примерно 14% пар — попадает в диапазон погрешности в 0,1–1,0 с. Здесь, как правило, достаточно слегка скорректировать границы вручную. Все зафиксированные грубые сбои (та самая доля в 8%) приходятся на ролики с откровенно низким качеством записи, заметным фоновым шумом и одновременным присутствием в кадре нескольких говорящих — то есть на условия, в которых модель не тренировалась.

Подсистема автоматизированного сбора датасета

Качественное обучение нейросети невозможно без репрезентативной коллекции видеозаписей с крупным планом лица говорящего [9], [10]. Чтобы исключить трудоёмкий ручной отбор, в комплексе предусмотрен модуль, который самостоятельно собирает подобные ролики на открытых видеосервисах. Опорой модуля служит свободная утилита yt-dlp [11], способная скачивать материал с большого спектра площадок. Из Python yt-dlp вызывается посредством стандартного модуля subprocess, которому передаются нужные опции: ограничение разрешения 480p (этого более чем достаточно для выделения области губ, при этом существенно экономится место на диске), путь сохранения, верхний предел по размеру и максимальная продолжительность одного ролика [12].

Высокоуровневая логика массового сбора реализована в отдельном скрипте download_training_data.py. Он обрабатывает заранее подготовленный список из 40 англоязычных поисковых запросов, охватывающих разнообразные сюжеты с одиночным говорящим: интервью, подкасты, лекции, обучающие материалы и обзоры. Для каждого запроса скрипт скачивает по три ролика. Между запросами выдерживается пятиминутная пауза — это снижает вероятность временной блокировки со стороны видеохостинга. Перед каждым новым запросом проверяется объём свободного места на диске: если он падает ниже 30 ГБ скрипт аккуратно завершает работу, сохраняя всё ранее загруженные файлы. Состав запросов специально подобран так, чтобы обеспечить вариативность условий съёмки, ракурсов, освещения, голосов и тем — без этого обучаемая модель не получит достаточной обобщающей способности.

Когда загрузка завершена, скачанные ролики проходят пакетную предобработку, которую запускает команда python -m av_sync.main preprocess <video_dir>. В ходе предобработки формируются серии кадров с губами, рассчитываются мел-спектрограммы, которые затем нарезаются под видеокадры. Итог записывается на диск в формате .npy с поддержкой memory-mapping. Такая схема хранения избавляет от необходимости держать весь датасет в RAM при дальнейшей работе. Файлы, прошедшие обработку ранее, пропускаются автоматически — это даёт возможность инкрементально дополнять корпус новыми записями. Если обработка отдельного ролика срывается (например, видеоконтейнер повреждён либо в кадрах не нашлось лица), скрипт пишет об этом в журнал и переходит к следующему файлу, не останавливая обработку остальных.

Интерфейсы пользователя

6.1. Графический интерфейс

Оконный интерфейс реализован на PyQt6 — официальной привязке Python к фреймворку Qt 6, который давно стал отраслевым стандартом для кроссплатформенных настольных приложений. Главное окно поделено на четыре функциональные зоны: панель выбора входных файлов (видео и аудио — через системные диалоги открытия файлов), панель прогресса с указанием текущей стадии обработки, область визуализации полученного результата и панель управления экспортом.

Визуальная зона ответа включает два связанных компонента: тепловую карту, которая отрисовывает матрицу косинусных сходств, и табличный перечень обнаруженных сегментов. Тепловую карту строит Matplotlib [13], встраиваемый в Qt-окно через FigureCanvasQTAgg. Горизонтальная ось отвечает за аудиовремя, вертикальная — за видеовремя, а цветовая шкала отображает значения косинусного сходства. Теоретически этот показатель варьируется в диапазоне от –1 до +1, но в типовом материале значимые значения сосредоточены примерно в диапазоне от –0,3 до +0,9, поэтому именно эти границы используются по умолчанию для контрастной отрисовки. Поверх тепловой карты накладываются траектории найденных сегментов в виде ломаных. В таблице сегментов представлены пять столбцов: видеостарт, видеоконец, аудиостарт, аудиоконец и оценка уверенности. Клик по строке таблицы переводит встроенный плеер на воспроизведение выбранного отрезка.

Ресурсоёмкие стадии обработки — подготовка видео, инференс сети и сборка итогового файла — вынесены в отдельный QThread, что предотвращает «зависание» основного потока GUI. Прогресс передаётся в главное окно посредством механизма сигналов Qt и отображается в виде процентной шкалы с подписью текущего этапа. При прерывании операции пользователем (нажатии кнопки «Отмена») во внутреннем потоке устанавливается флаг отмены, который регулярно проверяется во всех длительных циклах подсистем. Благодаря этому циклы корректно завершают работу и освобождают захваченные ресурсы.

6.2. Пакетный режим

Для удобства встраивания инструмента в автоматизированные конвейеры, предусмотрен запуск без графической оболочки. Весь цикл обработки поднимается одной командой python -m av_sync.main sync video.mp4 audio.wav output.mp4. На выходе сохраняется готовый файл по указанному пути, а в стандартный вывод поступают журнал работы и числовые показатели качества в формате JSON, что упрощает их подхват внешними инструментами. Стандартный код выхода (нулевой при удаче, иной — при ошибке) делает инструмент совместимым со скриптовыми оболочками Bash и PowerShell, утилитой Make и привычными системами непрерывной интеграции.

Эксплуатационные характеристики

Замеры производительности проводились на трёх показательных конфигурациях: рабочей станции с GPU NVIDIA RTX 3060, ноутбуке с процессором Intel Core i7-12700H без отдельного графического ускорителя и облачной виртуальной машине с 4 vCPU. Тестовая нагрузка во всех случаях была одна и та же — пара «видеоклип длиной 10 с в 720p плюс 15-минутная звуковая запись». На каждой конфигурации фиксировались две величины: полное время прохода и пик потребления памяти. Сводка результатов представлена в таблице 3.

Таблица 3 - Время обработки одной пары для различных пользовательских конфигураций

DOI: https://doi.org/10.60797/IRJ.2026.169.17.3

Этап	RTX 3060 (GPU)	Intel i7 (CPU)	VM 4 vCPU
Предобработка видео, с	3,2	3,5	4,1
Предобработка аудио, с	1,8	2,1	2,4
Инференс сети, с	1,4	23,8	47,2
Постобработка матчей, с	0,2	0,2	0,3
Сборка MP4 (FFmpeg), с	2,6	8,1	12,7
Полное время, с	**9,2**	**37,7**	**66,7**
Пик ОЗУ, ГБ	2,1	2,3	2,5
Пик видеопамяти, ГБ	3,4	–	–

При использовании отдельной видеокарты NVIDIA RTX 3060 полный проход одной пары укладывается в 9,2 с. При этом непосредственно прогон через сеть отнимает порядка 1,4 с, а основная часть времени уходит на подготовительные стадии и финальную сборку. Если же расчёт выполняется исключительно на CPU ноутбука Intel Core i7-12700H, то общее время вырастает до 37,7 с. Примерно 24 с из них тратятся на инференс: здесь даёт себя знать отсутствие того массового параллелизма по блокам, который GPU обеспечивает «бесплатно». В сценарии облачной виртуальной машины с 4 vCPU время обработки увеличивается до 66,7 с — это самый медленный из проверенных режимов, однако и в нём инструмент остаётся работоспособным. Что касается потребления памяти, верхняя планка ОЗУ во всех трёх замерах не превышает 2,5 ГБ. Диапазон 2,1–2,5 ГБ обусловлен разной кэш-стратегией с GPU и без него. Пиковое потребление видеопамяти остаётся в пределах 3,4 ГБ — такие требования вполне посильны даже для бюджетных потребительских ускорителей.

Заключение

В настоящей статье изложены архитектура и двухступенчатый алгоритм сопоставления, образующие основу программного комплекса автоматической синхронизации аудио- и видеопотоков по артикуляции говорящего. Сформулируем главные итоги.

1. Построена модульная архитектура комплекса, объединяющая шесть функциональных подсистем — подготовку видео, подготовку аудио, собственно синхронизацию, обучение и оценку, графический интерфейс и сбор обучающих данных, — которые связаны простыми типами данных и общим конфигурационным модулем. Принятая архитектура одинаково поддерживает работу как через оконный интерфейс, так и в пакетном режиме из командной строки и не требует изменения исходного кода при переключении между CPU и GPU.

2. Разработан и реализован двухступенчатый алгоритм сопоставления. Грубый этап выполняет пакетный расчёт эмбеддингов для пятисекундных блоков и собирает матрицу косинусных сходств. Этап постобработки включает медианную фильтрацию аудиопозиций, проверку их монотонности с интерполяционной заменой выбросов и слияние смежных совпадений в непрерывные сегменты. На тестовой подвыборке из 50 пар «короткое видео — длинная аудиозапись» алгоритм корректно локализует совпадение в 92,0% случаев со средней абсолютной ошибкой начала корректно найденного сегмента 71 мс.

3. Эксплуатационные испытания, проведённые в трёх характерных средах, дали следующие показатели: полный проход одной пары занимает 9,2 с при наличии массовой видеокарты и 37,7 с на CPU обыкновенного ноутбука; верхний предел по оперативной памяти ограничен 2,5 ГБ, по видеопамяти — 3,4 ГБ. Такая комбинация скорости и скромного потребления ресурсов делает разработанный инструмент пригодным для тех ниш, в которых громоздкие исследовательские реализации SyncNet, Wav2Lip и им подобные просто не работают: для повседневной работы журналистов и видеомонтажёров на личных машинах, для архивных учреждений, для дешёвых виртуальных серверов общедоступных хостингов.

Дальнейшее развитие комплекса видится в нескольких направлениях. Во-первых, обработка кадров, в которых одновременно говорят несколько человек. Во-вторых, добавление онлайн-режима, позволяющего синхронизировать потоки по мере их поступления, — это важно для систем видеоконференций. В-третьих, подключение к популярным редакторам нелинейного видеомонтажа через плагины.

Конфликт интересов

Не указан.

Рецензия

Рудой Е.М., ООО «ГК «Иннотех», Москва Российская Федерация

DOI: https://doi.org/10.60797/IRJ.2026.169.17.4

Conflict of Interest

None declared.

Review

Rudoi E.M., Innotech Group LLC, Moscow Russian Federation

DOI: https://doi.org/10.60797/IRJ.2026.169.17.4

Список литературы / References

1. Корнеев В.В. Специализированные аппаратные нейросетевые ускорители / В.В. Корнеев // Программная инженерия. — 2023. — Т. 14, № 1. — С. 3–11. — DOI: 10.17587/prin.14.3-11.

2. Roy A. SyncNet: Harmonizing Nodes for Efficient Learning / A. Roy, B.S. Gyanchandani, J. Sahu [et al.] // 2024 International Conference on Software, Telecommunications and Computer Networks (SoftCOM). — Split, Croatia, 2024. — P. 1–6. — DOI: 10.23919/SoftCOM62040.2024.10721994.

3. Xueming Q. Design and Implementation of a Wav2Lip-based Digital Human Framework for Intelligent Power Customer Service / Q. Xueming [et al.] // 2025 IEEE 3rd International Conference on Electrical, Automation and Computer Engineering (ICEACE). — Changchun, China, 2025. — P. 892–896. — DOI: 10.1109/ICEACE67491.2025.11439472.

4. Virkkunen A. Investigating the Clusters Discovered By Pre-Trained AV-HuBERT / A. Virkkunen, M. Sarvaš, G. Huang [et al.] // ICASSP 2024 — 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). — Seoul, Republic of Korea, 2024. — P. 11196–11200. — DOI: 10.1109/ICASSP48485.2024.10447434.

5. Long Y. Enhancing Educational Content Matching Using Transformer Models and InfoNCE Loss / Y. Long, D. Gu, X. Li [et al.] // 2024 IEEE 7th International Conference on Information Systems and Computer Aided Education (ICISCAE). — Dalian, China, 2024. — P. 11–15. — DOI: 10.1109/ICISCAE62304.2024.10761438.

6. Lei X. Design and Implementation of a Real-Time Video Stream Analysis System Based on FFMPEG / X. Lei, X. Jiang, C. Wang // 2013 Fourth World Congress on Software Engineering. — Hong Kong, China, 2013. — P. 212–216. — DOI: 10.1109/WCSE.2013.38.

7. Куликов Р.И. Информационная система на основе нейросети для автоматической генерации субтитров к видео / Р.И. Куликов // Профессиональные коммуникации в научной среде — фактор обеспечения качества исследований : материалы XIII Всерос. науч.-практ. конф., Альметьевск, 16 апреля 2024 г. — Санкт-Петербург : ООО Издательский дом «Сциентиа», 2024. — С. 333–336.

8. Wang Z. Ocular Feature Extraction for Eye Movement Analysis and Neurological Dysfunction Diagnosis / Z. Wang [et al.] // 2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC). — Vienna, Austria, 2025. — P. 7626–7631. — DOI: 10.1109/SMC58881.2025.11342937.

9. Слядников Е.Е. Модель, алгоритм и программная реализация обработки данных для видеокодека / Е.Е. Слядников // Доклады Томского государственного университета систем управления и радиоэлектроники. — 2012. — № 1-1(25). — С. 65–70.

10. Желтоухов М.О. Система автоматической подготовки датасетов с последующим обучением нейросети для задач компьютерного зрения / М.О. Желтоухов, В.С. Тарасов // Сборник студенческих научных работ факультета компьютерных наук ВГУ : сб. ст. : в 2 ч. / под ред. Д.Н. Борисова. — Воронеж : Воронежский государственный университет, 2019. — № 13, ч. 2. — С. 95–100.

11. B G. Social Media Content Generation: An AI-Based System for Automated Viral Short-Form Video Creation / G. B, K. Somasundaram // 2025 IEEE 2nd International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS). — Bangalore, India, 2025. — P. 1–9. — DOI: 10.1109/ICITEICS64870.2025.11341675.

12. Антипова С.А. Разработка системы контроля доступа на основе распознавания лиц / С.А. Антипова // Программные продукты и системы. — 2021. — № 2. — С. 245–256. — DOI: 10.15827/0236-235X.134.245-256.

13. Ari N. Matplotlib in python / N. Ari, M. Ustazhanov // 2014 11th International Conference on Electronics, Computer and Computation (ICECCO). — Abuja, Nigeria, 2014. — P. 1–6. — DOI: 10.1109/ICECCO.2014.6997585.

Список литературы на английском языке / References in English

1. Korneev V.V. Spetsializirovannye apparatnye neyrosetevye uskoriteli [Specialized hardware neural network accelerators] / V.V. Korneev // Programmnaya inzheneriya [Software Engineering]. — 2023. — Vol. 14, № 1. — P. 3–11. — DOI: 10.17587/prin.14.3-11. [in Russian]

2. Roy A. SyncNet: Harmonizing Nodes for Efficient Learning / A. Roy, B.S. Gyanchandani, J. Sahu [et al.] // 2024 International Conference on Software, Telecommunications and Computer Networks (SoftCOM). — Split, Croatia, 2024. — P. 1–6. — DOI: 10.23919/SoftCOM62040.2024.10721994.

3. Xueming Q. Design and Implementation of a Wav2Lip-based Digital Human Framework for Intelligent Power Customer Service / Q. Xueming [et al.] // 2025 IEEE 3rd International Conference on Electrical, Automation and Computer Engineering (ICEACE). — Changchun, China, 2025. — P. 892–896. — DOI: 10.1109/ICEACE67491.2025.11439472.

4. Virkkunen A. Investigating the Clusters Discovered By Pre-Trained AV-HuBERT / A. Virkkunen, M. Sarvaš, G. Huang [et al.] // ICASSP 2024 — 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). — Seoul, Republic of Korea, 2024. — P. 11196–11200. — DOI: 10.1109/ICASSP48485.2024.10447434.

5. Long Y. Enhancing Educational Content Matching Using Transformer Models and InfoNCE Loss / Y. Long, D. Gu, X. Li [et al.] // 2024 IEEE 7th International Conference on Information Systems and Computer Aided Education (ICISCAE). — Dalian, China, 2024. — P. 11–15. — DOI: 10.1109/ICISCAE62304.2024.10761438.

6. Lei X. Design and Implementation of a Real-Time Video Stream Analysis System Based on FFMPEG / X. Lei, X. Jiang, C. Wang // 2013 Fourth World Congress on Software Engineering. — Hong Kong, China, 2013. — P. 212–216. — DOI: 10.1109/WCSE.2013.38.

7. Kulikov R.I. Informatsionnaya sistema na osnove neyroseti dlya avtomaticheskoy generatsii subtitrov k video [Information system based on a neural network for automatic generation of subtitles for video] / R.I. Kulikov // Professional'nye kommunikatsii v nauchnoy srede — faktor obespecheniya kachestva issledovaniy : materialy XIII Vseros. nauch.-prakt. konf., Al'met'yevsk, 16 aprelya 2024 g. [Professional communications in the scientific environment — a factor in ensuring the quality of research : pProceedings of the XIII All-Russian Scientific-Practical Conference, Almetyevsk, April 16, 2024]. — Saint Petersburg : LLC Publishing House «Stsientia», 2024. — P. 333–336. [in Russian]

8. Wang Z. Ocular Feature Extraction for Eye Movement Analysis and Neurological Dysfunction Diagnosis / Z. Wang [et al.] // 2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC). — Vienna, Austria, 2025. — P. 7626–7631. — DOI: 10.1109/SMC58881.2025.11342937.

9. Slyadnikov E.E. Model', algoritm i programmnaya realizatsiya obrabotki dannykh dlya videokodeka [Model, algorithm and software implementation of data processing for a video codec] / E.E. Slyadnikov // Doklady Tomskogo gosudarstvennogo universiteta sistem upravleniya i radioelektroniki [Reports of Tomsk State University of Control Systems and Radioelectronics]. — 2012. — № 1-1(25). — P. 65–70. [in Russian]

10. Zheltoukhov M.O. Sistema avtomaticheskoy podgotovki datasetov s posleduyushchim obucheniem neyroseti dlya zadach komp'yuternogo zreniya [System for automatic dataset preparation with subsequent neural network training for computer vision tasks] / M.O. Zheltoukhov, V.S. Tarasov // Sbornik studencheskikh nauchnykh rabot fakul'teta komp'yuternykh nauk VGU : sb. st. : v 2 ch. [Collection of student scientific works of the Department of Computer Science of VSU : collection of articles : in 2 parts] / ed. by D.N. Borisov. — Voronezh : Voronezh State University, 2019. — № 13, pt. 2. — P. 95–100. [in Russian]

11. B G. Social Media Content Generation: An AI-Based System for Automated Viral Short-Form Video Creation / G. B, K. Somasundaram // 2025 IEEE 2nd International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS). — Bangalore, India, 2025. — P. 1–9. — DOI: 10.1109/ICITEICS64870.2025.11341675.

12. Antipova S.A. Razrabotka sistemy kontrolya dostupa na osnove raspoznavaniya lits [Development of an access control system based on face recognition] / S.A. Antipova // Programmnye produkty i sistemy [Software Products and Systems]. — 2021. — № 2. — P. 245–256. — DOI: 10.15827/0236-235X.134.245-256. [in Russian]

13. Ari N. Matplotlib in python / N. Ari, M. Ustazhanov // 2014 11th International Conference on Electronics, Computer and Computation (ICECCO). — Abuja, Nigeria, 2014. — P. 1–6. — DOI: 10.1109/ICECCO.2014.6997585.