Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский
Академический университет Российской академии наук»
Центр высшего образования

Кафедра математических и информационных технологий

Чаркин Константин Эдуардович

Автоматическая генерация конспекта по видеокурсу

Магистерская диссертация

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Омельченко А. В.

Научный руководитель:
Вяххи Н. И.

Рецензент:
к. т. н., доцент Дробинцев П. Д.

Санкт-Петербург
2017

SAINT-PETERSBURG ACADEMIC UNIVERSITY
Higher education centre

Department of Mathematics and Information Technology

Konstantin Charkin

Automatic video course summarization

Graduation Thesis

Admitted for defence.
Head of the chair:
professor Alexander Omelchenko

Scientific supervisor:
Nikolay Vyahhi

Reviewer:
assistant Pavel Drobintsev

Saint-Petersburg
2017

Оглавление

Введение

С каждым годом онлайн-обучение становится все более популярным. Это связано с преимуществами онлайн-обучения перед традиционным подходом в виде классического очного высшего образования. Обычно к плюсам онлайн-обучения относят:

- Возможность выбрать удобное время для обучения
- Возможность выбрать план обучения и нагрузку
- Проще совмещать с какой-либо другой деятельностью
- Нет ограничений на географическое местоположение
- Обычно значительно дешевле
- и многие другие

Основная форма онлайн-обучения – это массовый открытый онлайн-курс (MOOK). *MOOK* – это онлайн-курс нацеленный на неограниченное участие и доступ через Интернет. Обычно он состоит из теории в виде видеолекций или текста, множества задач с автоматичекой проверкой либо задач с рецензированием (когда студенты проверяют решения друг друга), и набора ссылок на дополнительные материалы. Зачастую онлайн-курсы включают форумы, на которых студенты могут общаться друг с другом либо с преподавателями курса, для того, чтобы максимально быстро получить обратную свзязь при возникновении затруднений при прохождении курса.

Stepik.org [28] – одна из наиболее популярных платформ онлайн-обучения, с преимущественно рускоязычным контентом. Онлайн-курсы, в которых основная часть теории представлениа в виде видеолекции называются *видеокурсами*. На stepik.org 352 открытых онлайн-курса на которые записалось больше 10 человек, из них 114 (32%) – это видеокурсы. Также, на stepik.org 75% активности (по количеству отправленных решений задач) генерируется в видеокурсах. Это говорит о том, что видеокурсы – это полулярный вид онлайн-обучения.

С другой стороны существует множество исследований [4] [9] [23] [37], которые показывают, что видеокурсы – это хороший способ обучения. Основные причиты этого в том, что, во-первых, из-за постоянного контакта с преподавателем у студентов выше концентрация внимания, в каком-то смысле такой формат больше похож на очные лекции, во-вторых, при просмотре видео студенты меньше устают, вследствие этих причины изучаемый материал лучше усваивается.

Но у видеокурсов существуют и некоторые проблемы. Первая проблема заключается в том, что если при прохождении курса, например при решении задачи, пользователю понадобилась некоторая информация (математическая формула, определение какого-либо понятия, команда bash и т.д.), которая озвучивалась где-то ранее в данном курсе, но он точно не помнит где, то единственный способ поиска заключается в последовательном просмотре всех прошедших видео. Но такой способ неудобен, малоэффективен, требует большого количества времени. Также, если пропустить нужную информацию, то придется начинать заново. Вторая проблема заключается в том, что если спустя какое-то время, например полгода или год, пользователю понадобится полученная в курсе информация (на собеседовании на новую работу, на экзамене или по работе), и ему нужно будет в максимально короткий срок повторить пройденный материал, то имея доступ только к видеозаписям сделать это быстро не получится. Основной подход при решении первой проблемы – это предоставить пользователю удобный механизм поиска по видео. В своих исследованиях W.H.Chang et al. [7] [8] описывали систему для краткого представления видео на основе ключевых слов, которая позволяла быстро навигироваться по исходными видеолекциям. Coursera [10] – одна из крупнейших платформ для онлайн-обучения, предоставляет пользователям то, что они называют *интерактивный текст видеоматериала*. Под видеолекцией располагается текст данной лекции (рис. 1), в котором каждое слово является ссылкой на момент видео, который соответствует данному слову. Данные подходы в большей степени нацелены на решения проблемы навигации и поиска. На платформе онлайн-обучения Coursera есть также

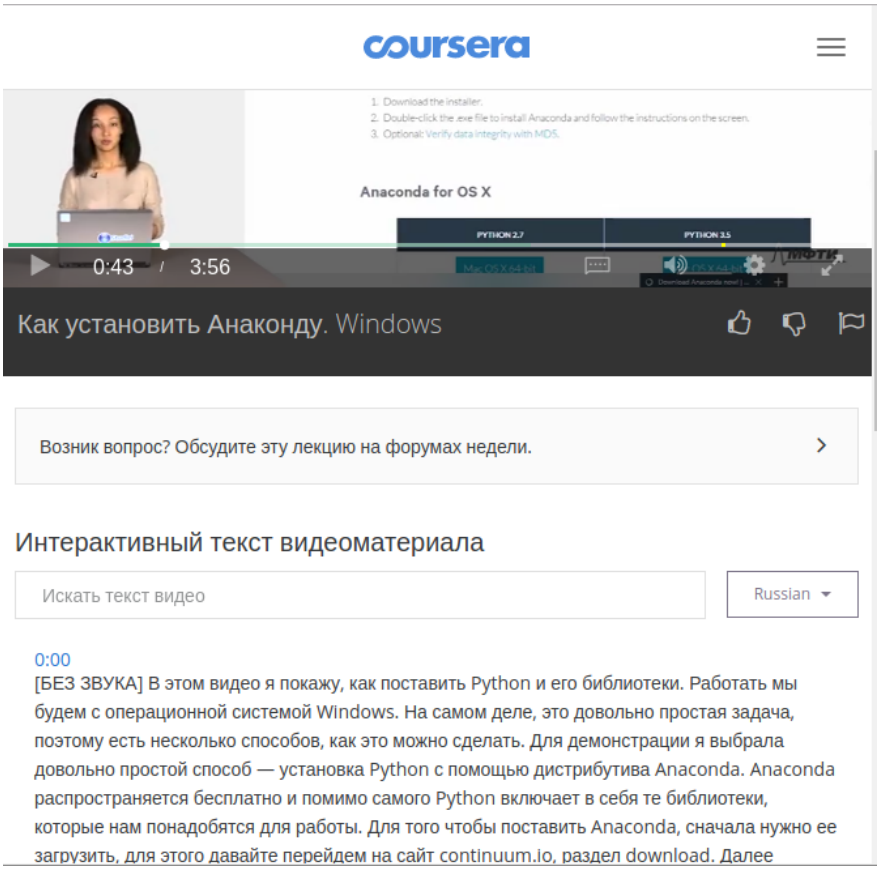


Рис. 1: Интерактивный текст видеоматериала на платформе онлайн-обучения Coursera

возможность скачать тексты видеолекцией в формате ".txt". Но использовать их для повторения материала сложно, так как данные тексты не имеют форматирования, в них нет графических материалов, математических формул и многих других вещей, которые могли бы помочь решить обозначенную проблему.

С другой стороны очевидно, что при наличии у студентов конспекта в классическом понимании этого слова, т.е. некоторого эквивалента видеокурса в форме текста, картинок, схем, таблиц, формул и т.д., то обе обозначенные проблемы будут решены. Так как текст удобен для поиска, а если в тексте также будут ссылки на исходные видео, то данный текст можно будет для навигации по видеокурсу. Конспект также решает проблему с необходимостью повторить пройденный материал в максимально короткие сроки, так как это и есть главное предназначение конспекта.

Идеальным решением было бы просить преподавателей предоста-

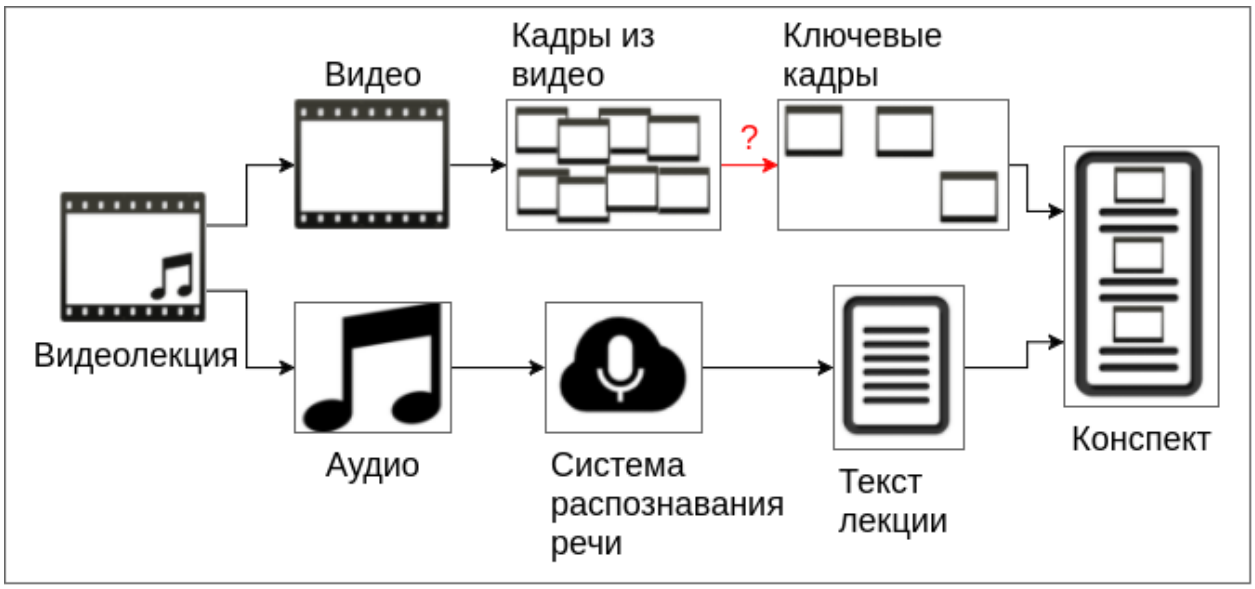


Рис. 2: Схема генерации конспекта по видеолекции

вить конспект курса, который они читают. Но одной из задач, решаемых платформой stepik.org является создание удобного конструктора курсов, чтобы преподаватели приходили на платформу, создавали курсы, делились ими с остальными пользователями, и чтобы это происходило полностью без участия команды stepik.org. Причем для преподавателей это должно быть максимально просто, чтобы как можно больше преподавателей завершало создание курсов. В то же время, написание конспекта – это трудная задача, следовательно, просьба преподавателям предоставить конспект их курсов усложняет для них процесс создания этих курсов, поэтому от данного варианта пришлось отказаться.

В данной работе представлена система по автоматической генерации конспекта по видеокурсу на платформе онлайн-обучения stepik.org. Схема генерации представлена на рис. 2. *Ключевые кадры* – это минимальный набор кадров видео такой, что в нем содержится вся полезная визуальная информация видеолекции, которую преподаватель хотел донести до студентов.

Последовательность действий для генерации конспекта:

- Выделить из видеолекции аудио- и видеоряд
- Из аудио, с помощью системы распознавания речи, получить текст лекции

- Разбить видео на множество кадров
- Выбрать из него множество ключевых кадров
- Собрать текст и ключевые кадры в единый документ, с учетом их положения в лекции

Одной из основных идей по генерации конспекта является предположение о том, что набор ключевых кадров настолько мал, что все такие кадры можно поместить в конспект. Или другими словами, что почти все кадры видео либо не содержат полезной информации вовсе, либо полезная информация, расположенная на некотором кадре, также дублируется на множестве других кадров видео. Проблема заключается в том, что чтобы решить задачу по извлечению ключевых кадров в общем случае, нужно "понимать смысл" информации на кадрах, чего никакой алгоритм сделать не может. Поэтому, для решения данной задачи делаются дополнительные предположения, которые позволяют извлекать ключевые кадры, на основе метрик, которые можно посчитать автоматически, без необходимости прибегать к экспертному мнению. При построении системы автоматической генерации конспекта это наиболее сложная из решаемых задач и ей уделена значительная часть работы. Еще одна проблема описанного подхода заключается в том, что полученный таким образом конспект решает описанные проблемы не в полной мере, так как полный текст лекции не самым лучшим образом подходит для быстрого повторения материала. Когда конспект пишется человеком, студентом или преподавателем, то информация, озвученная в лекции перерабатывается таким образом, чтобы представить ту же информацию, но в более компактном виде. Для этого могут использоваться сводные таблицы, иллюстрации, различные специальные нотации и другие приемы. По очевидным причинам, сделать это автоматически не представляется возможным. Поэтому в данной работе предлагается предоставить пользователям кроме доступа к чтению полученного конспекта, также и доступ к его редактированию. Тогда со временем пользователи самостоятельно доведут автоматически сгенерированную предварительную версию до конечного состояния.

Таким образом, целью данной работы является предоставить пользователям платформы онлайн-обучения stepik.org конспекты курсов, расположенных на данной платформе. Для этого нужно будет решить ряд задач:

- Реализовать извлечение ключевых кадров из видеолекций
- Реализовать получение текста лекции по ее аудиоряду
- Собрать полученную информаци в единый документ
- Предоставить пользователям доступ к чтению, скачиванию и совместному редактированию полученных документов
- Оформить систему по генерации конспектов в виде удобного для преподавателей и команды stepik.org инструмента

Данная работа состоит из пяти глав. В главе 1 описаны основные возможности платформы stepik.org и смежные понятия. Глава 2 посвящена извлечению ключевых кадров. В ней произведен обзор существующих инструментов для решения данной задачи, обзор актуальных исследований по данной и смежным темам. Разработан и реализован алгоритм извлечения ключевых кадров для задачи генерации конспектов по курсам на платформе stepik.org. Также произведено сравнение различных вариаций разработанного алгоритма с существующими решениями. Глава 3 содержит описание выбранного способа получения текста лекции по ее аудиоряду. В главе 4 описана реализация предоставления пользователям доступа к чтению, скачиванию и совместному редактированию сгенерированных конспектов. Глава 5 содержит общее описание архитектуры сервиса по генерации конспектов. После основной части следует заключение, в котором изложены полученные результаты и указано их соотношение с общей целью и конкретными задачами, сформулированными во введении.

1. Платформа stepik.org

1.1. Общие сведения

Stepik.org – это облачная платформа, предназначенная для создания и распространения интерактивного образовательного контента, с поддержкой различных типов задач с автоматической проверкой и обратной связью в режиме реального времени. На stepik.org содержится больше ста онлайн-курсов, большая часть которых технической направленности. Основная часть контента на русском языке. Главные идеи, которых придерживается команда stepik.org при разработке платформы: свободный доступ (весь контент распространяется под открытой лицензией Creative Commons), интерактивные задания, персонализированное обучение и пользовательский контент. Последний пункт говорит о том, что одним из основных направлений деятельности является создание удобного и максимально простого конструктора курсов и уроков, чтобы любой пользователь, без специальных навыков и знаний, мог прийти на платформу, создать качественный контент и поделиться им с другими пользователями.

1.2. Терминология

Наиболее крупной структурной единицей контента является *курс*. Курс состоит из нескольких (обычно от 3 до 7) *модулей*. В курсах с дедлайнами модуль обычно рассчитан на одну неделю. Каждый модуль входит ровно в 1 курс.

Модуль в свою очередь состоит из *уроков*. Уроки это самостоятельная единица контента, посвященная некоторой законченной теме. Уроки могут существовать как отдельно от курсов, так и входить в множество различных курсов. В модуль обычно входит от пяти до семи уроков.

Уроки делятся на *шаги*. Шаг – это либо теория в форме текста или видео, или одна из примерно двадцати типов задач, среди которых есть как задачи с автоматичекой проверкой (задачи на написание

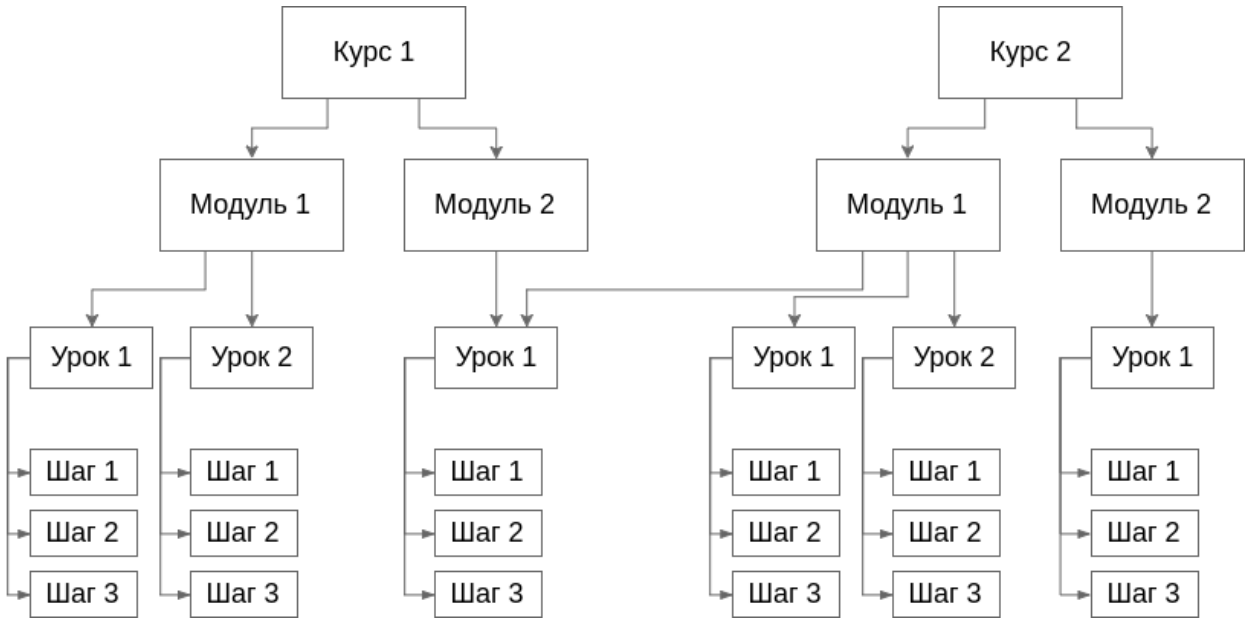


Рис. 3: Схема структуры контента на платформе stepik.org

кода, задачи с выбором одного или нескольких ответов и многие другие), так и задачи на написание эссе, которые пользователи проверяют друг у друга по сформулированным преподавателем критериям. Длительность видео в шаге обычно не превышает 5-15 минут. Каждый шаг входит ровно в один урок. Урок состоит не более чем из шестнадцати шагов. Общая схема структуры контента представлена на рис. 3

2. Извлечение ключевых кадров

2.1. Обзор готовых решений

Существует множество инструментов для решения задачи извлечения ключевых кадров. В данной главе будут рассмотрены 3 таких инструмента, среди которых:

- ffprobe [14]
- Shotdetect [25]
- PySceneDetect [21]

Все они работают по схожим принципам и используют один из двух подходов: анализ различия между соседними кадрами путем подсчета суммы расстояний между соответствующими пикселями либо сравнение яркости каждого кадра с пороговым значением. Второй подход основан на предположении, что переходы между сценами в видео происходят через переход к черному или почти черному кадру. Существует сравнение этих инструментов [1], при котором было выбрано 3 видеозаписи, каждое видео было размечено вручную и проведен анализ результатов по каждому инструменту. Результат сравнения представлен на рис. 4. На графиках представлены кривые вида полнота/точность.

Также, для каждой пары видео/инструмент посчитана площадь под данными кривыми и представлена в табл. 1. Значение данного параметра лежит в интервале $[0, 1]$ (чем больше, тем лучше) и является агрегированной характеристикой качества классификации.

Таблица 1: Сравнение различных инструментов для извлечения ключевых кадров, площадь под кривой полнота/точность.

видео	ffprobe	shotdetect	PySceneDetect
1	0.97	0.97	0.86
2	0.99	0.99	0.99
3	0.50	0.45	0.46

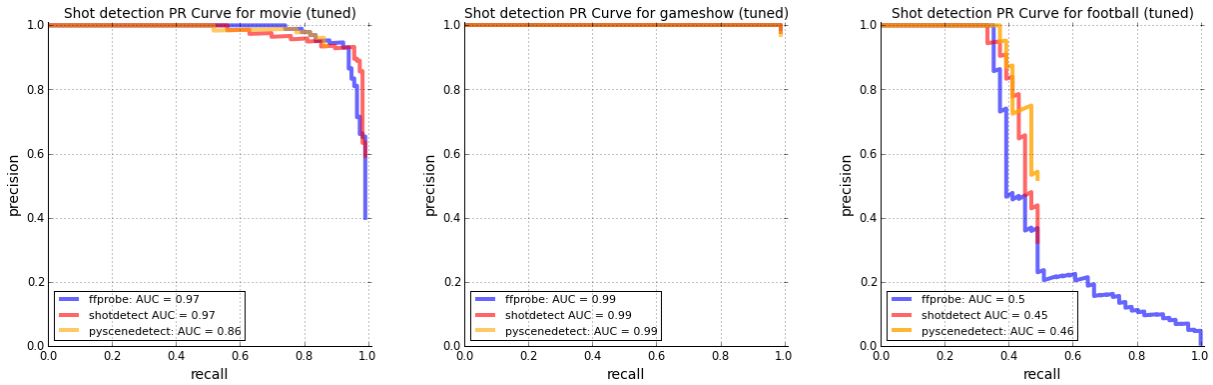


Рис. 4: Сравнение различных инструментов для извлечения ключевых кадров, кривые полнота/точность.

Нетрудно заметить, что на всех трех видео все представленные решения показывают схожие результаты. Поэтому в данной работе разработанные алгоритмы сравнивались с PySceneDetect, так как данная библиотека имеет более удобный, по сравнению с альтернативами, интерфейс. Она написана на языке программирования Python, который также использовался для реализации извлечения ключевых кадров в данной работе.

2.2. Обзор литературы

В последние десятилетия технологии связанные с обработкой цифрового видео развивались довольно большими темпами. Благодаря им, становится очень просто записывать огромное количество видео. Существует много различного видеоконтента такого как новости, фильмы, спорт и т.д. Также, тысячи камер видеонаблюдения можно найти в различных общественных местах. Все это приводит к трудностям, связанным с анализом этих данных в режиме реального времени. Кроме того, хранение огромного количества видеоданных также вызывает сложности. Все это приводит к необходимости создания инструментов которые бы решали подобного рода проблемы. Основная идея таких инструментов состоит в выделении из видео небольшого количества информации такого, чтобы из него была понятна большая часть происходящего на видео.

Также существует довольно большое количество исследований, посвященных созданию алгоритмов и инструментов на их основе [35, 31, 18, 6, 26, 3, 24, 22, 30] и множество других, описанных далее в данной работе, для решения задач краткого представления видео, извлечения ключевых кадров и смежных проблем. Довольно подробный обзор существующих подходов и методов представлен в работе Muhammad Ajmal а его коллег [33], где они также привели классификацию техник для решения задачи краткого представления видео. Они выделяют шесть основных направлений:

- кластеризация
- на основе анализа признаков
- на основе анализа событий
- на основе разбиения на сцены
- на основе анализа траекторий
- мозаичный метод

Суть большинства из них понятна из названия, за исключением последнего. Мозаичный метод применяется для анализа видео, в которых встречаются панорамные съемки. Данный метод позволяет получить из такого панорамного видео один большой кард, на котором будет изображена вся панорама. Затем, каждый из этих подходов делится на более мелкие, например в методе, основанном на анализе признаков могут использоваться такие признаки как движение, цвет, аудио информация, жесты и др. Также в этом исследовании делаются выводы о том, какие методы стоит применять в зависимости от типа решаемой задачи. Особенность классификации такой классификации заключается в том, что при проведении каких-либо исследований в данной области, либо при решении каких практических задач, всегда сразу несколько методов, которые находятся в разных частях иерархической классификации, предложенной в [33].

Более общими подходами в классификации методов краткого представления видео является разделения их по тому, какая задача решается и по тому, как подходить к анализу видео. В первом случае все исследования можно разделить на те, которые решают задачу в общем и в частном случае. При решении задачи в общем случае про анализируемые видео не делается никаких предположений, кроме тех, которые присуще любому видео. К таким относится предположение о том, что большинство кадров дублируют информацию, которая есть на других кадрах видео. Или другими словами, что обычно, при переходе от одного кадра к следующему изображение меняется не сильно. В видео это нужно для того, чтобы оно казалось единым целым, а не набором отдельных, сменяющихся, слабо связных кадров. Примерами таких исследований являются [16, 20].

При решении задачи в частном случае активно используется некоторое априорное знание про анализируемые видео. Например в своем исследовании Shanon X. Ju с коллегами [29] анализировали видеозаписи лекции, при которых в аудитории при помощи проектора слайды показывались на экране, лектор читал лекции, периодически переключая слайды и показывая руками на кадрах что-то, тем самым закрывая часть слайда. Ahmet Ekin с коллегами в своем исследовании [12] проводили анализ трансляций футбольных матчей с целью автоматического определение интересных моментов. Для этого были сделаны предположения о том, что интересные моменты происходят в основном около ворот, а также, что большую часть времени в кадре преобладает один цвет (цвет травы на поле), и что около ворот есть определенного вида разметка на поле.

По тому, как подходить к анализу видео, а точнее по тому, как смотреть на видео можно выделить два подхода. В первом случае видео представляется как множество (неупорядоченный набор) кадров, и в этом случае использует кластеризация, предварительно произведя некоторые преобразования с исходными кадрами, для извлечения ”хороших” признаков, на основании которых можно проводить кластеризацию. Затем, из каждого кластера выбирается определенное число

кадров, обычно либо один, либо пропорционально размеру кластера, и утверждается, что эти кадры являются ключевыми. При таком подходе теряется информация о положении кадров во времени. Делается в случаях, когда есть основания полагать, что в разные моменты времени в видео будут похожие кадры. Например, новостные передачи, в которых ведующий новостей появляется в кадре между репортадами и кластеризация позволяет добавить во множество ключевых кадров только один кадр с ведущим. Такой подход использовался в следующих исследованиях [2, 13, 20].

Альтернативным подходом является представление видео как последовательность (упорядоченный набор) кадров. В этом случае анализируются переходы между кадрами, т.е. то, как изменяется изображение от кадра к кадру. Дальше, по полученной информации, видео разбивается на интервалы. Разбиение обычно происходит по моментам времени, на которых либо происходят значительные, резкие изменения, либо если обнаруживается переход к черному кадру, такой прием часто используется для перехода между различными сценами. Далее считается, что внутри каждого интервала видео меняется слабо, т.е. если мы включим во множество ключевых кадров какого-либо кадр (обычно один из первых или один из последних), то он будет отражать происходящее внутри данного интервала и других кадров из него включать во множество ключевых не нужно. Примерами исследований с таким подходом являются [18, 38].

Также существуют исследования, в которых пытаются объединить два последних подхода, чтобы с одной стороны не терять информацию о временной составляющей положения кадров, а с другой получить плюсы кластеризации в виде большей точности ввиду того, что при кластеризации меньше дублей попадает во множество ключевых кадров. Подобный подход в своем исследовании продемонстрировали Yihong Gong и Xin Liu [17]. Им удалось представить временные и пространственные признаки изображений в едином векторном пространстве, в котором в дальнейшем и проводился анализ кадров, для краткого представления видео и поиска изображений в базе данных.

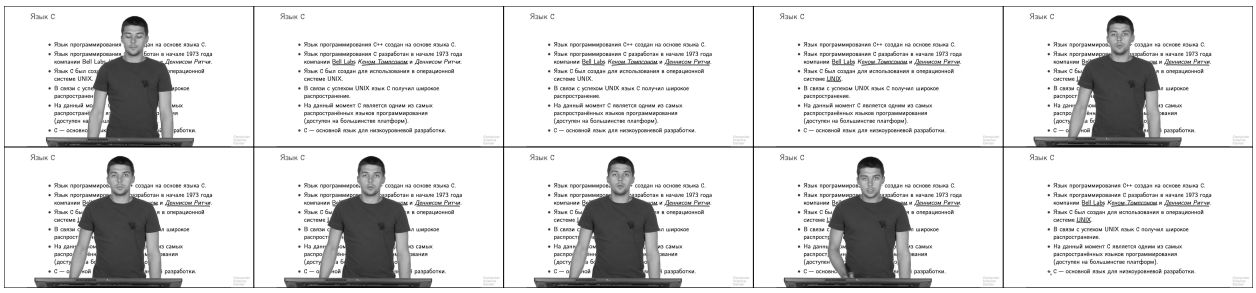


Рис. 5: Пример видеолекции для извлечения ключевых кадров

2.3. Реализация извлечения ключевых кадров

Реализация извлечения ключевых кадров происходила в два этапа. Сначала был разработан и реализован максимально простой вариант – наивный алгоритм. Это требовалось с одной стороны для того, чтобы проверить различные идея и максимально быстро получить результат, который можно оценить. А с другой – для того, чтобы при разработке более сложных версий было с чем сравнивать качество полученного результата. Так как готовые решения в виде PySceneDetect для этой цели подходят слабо, т.к. это инструмент для решения задачи в общем случае, а при разработке алгоритмов для извлечения ключевых кадров активно использовались априорные знания об анализируемых видео.

На втором этапе были оценены сильные и слабые стороны наивного алгоритма, и следующая версия разрабатывалась с целью оставить и развить сильные стороны и устранить слабые.

Также при разработке алгоритмов для решения поставленных задач в данной работе использовался подход, при котором видео рассматривается как упорядоченная последовательность кадров с анализом изменений происходящих от кадра к кадру. Такой подход был выбран, т.к. как уже было сказано ранее, подход с кластеризацией отказывается от знаний о порядке кадров для того, чтобы решить некоторые проблемы с дублированием кадров в результирующем множестве ключевых кадров. Но суть решаемой задачи такова, что такие ситуации в анализируемых видео встречаются довольно редко, и нет смысла использовать подход с кластеризацией.

Идея наивного алгоритма очень проста. Для начала посчитаем Δ –

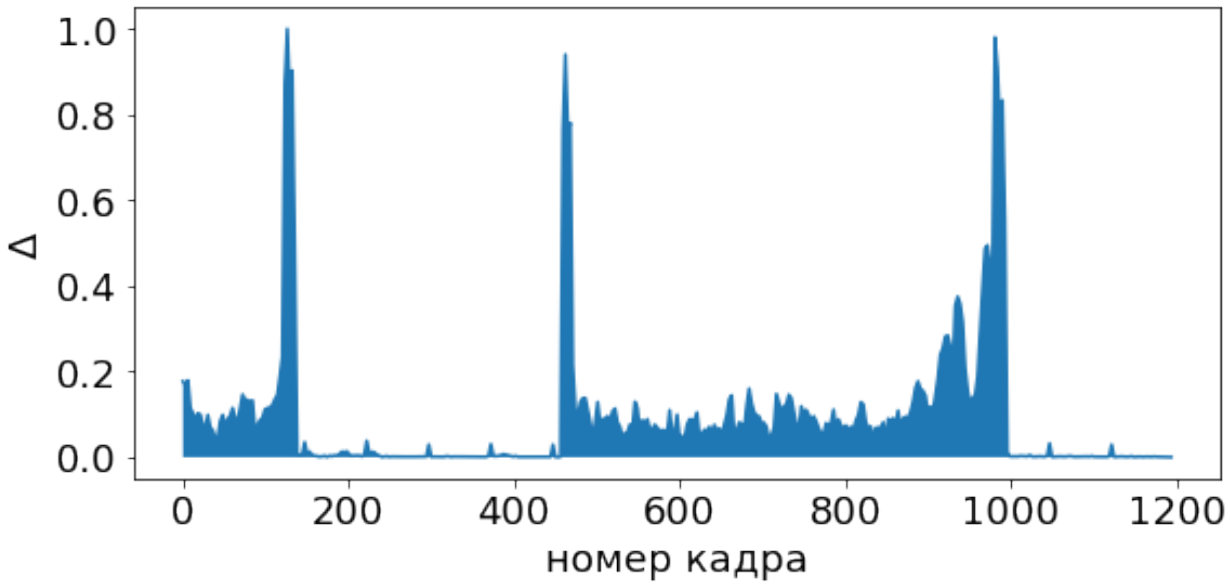


Рис. 6: График зависимости различия между соседними кадрами от номера кадра

разница между каждой парой соседних кадров, по формуле:

$$\Delta_k = \sum_{i=1}^{h} \sum_{j=1}^{w} |F_k[i,j] - F_{k+1}[i,j]| \tag{1}$$

где Δ_k – разница между k и $k+1$ кадрами, h – высота кадра в пикселях, w – ширина кадра в пикселях, F_k и F_{k+1} – k-й и $(k+1)$-й кадры соответственно, $F_k[i,j]$ - значение цвета пикселя в строке i и столбце j. Т.е. по сути это сумма расстояний между соответствующими пикселями.

Рассмотрим пример видеолекции, раскадровка (последовательность кадров через определенные промежутки времени, из которой понятно, что из себя представляет видео) которой представлена на рис. 5. Из нее мы видим, что в начале леции преподаватель находился в кадре, через некоторое время он перестал находится в кадре, а вместе него на видео присутствовал слайд с текстом, на котором преподаватель мог делать некоторые пометки. Потом опять появился преподаватель, затем опять слайд и видео закончилось. Если для этого видео посчитать Δ по формуле 1, то получится нечто похожее на то, что изображено на рис. 6. И в данном случае информации на данном графике достаточно для того, чтобы понять какие кадры нужно включить во множество ключевых

кадров. Пики около 170, 450 и 1000 кадра соответствуют кадрам, на которых преподаватель появляется или исчезает из кадра. Интервалы, на которых происходят значительные изменения, 0 по 170 и с 450 по 1000 кадры, соответствуют моментам времени, во время которых преподаватель находился в кадре. Т.к. он двигается, говорит, жестикулирует и производит другие действия, которые вносят различия в соседние кадры. С другой стороны, на интервалах с 170 по 150 и с 1000 до конца видео изменений почти не происходит, и эти интервалы соответствуют моментам, когда в кадре находился только слайд. Таким образом, для того, чтобы определить множество ключевых кадров, достаточно определить интервалы со слайдами и из каждого взять по одному кадру ближе к концу интервала, чтобы на нем были все пометки, которые по ходу лекции делал преподаватель.

Данный подход был реализован, и на видео, аналогичных рассмотренному (т.е. на таких, на которых в каждый момент времени в кадре находится либо преподаватель, либо слайд с полезной информацией) показывает хорошее качество извлечения ключевых кадров. Но такой подход перестает работать в случае, когда в кадре одновременно находится и преподаватель и полезная информация. В этом случае, за изменениями, вносимыми преподавателем на графике $\Delta(k)$ не видно переключений слайдов, т.к. абсолютные изменения от преподавателя значительно больше, чем смена одного текста на другой.

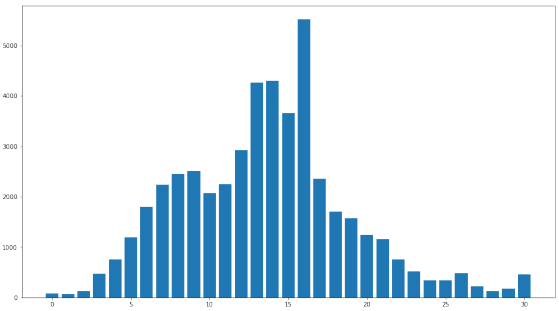
Для решения этой проблемы было решено сделать следующее: при подсчете Δ_k, перед применением формулы 1, используя распознавание людей на изображениях, найдем H_k – область (один или несколько прямоугольников) кадра F_k, соответствующая людям в данном кадре. Аналогично H_{k+1} для кадра F_{k+1}. После чего найдем объединение данных областей $H = H_k \cup H_{k+1}$, далее, на каждом кадре область соответствующую H закрасим черным цветов. Теперь, при подсчете разницы по формуле 1, наличие преподавателя в кадре будет влиять значительно меньше.

После применения поиска людей в кадре данный алгоритм стал работать значительно лучше на видео, на которых в кадре одновременно

находится и преподаватель и полезная информация (сравнение различных реализаций и PySceneDetect можно найти в следующем разделе). Но также оставалось некоторое количество проблем. Первая проблема – это то, каким образом происходит подсчет разницы между кадрами. При вычислении Δ по формуле 1 учитывается вся возможная разница, которая только может быть учтена. Но цель алгоритма на данном этапе, найти такие кадры, на которых происходят значительные изменения с точки зрения пользователя, смотрящего данную видеолекцию. А формула 1 с одной стороны учитывает разницу, которую человек вообще может не замечать, такую как шум, артефакты сжатия, незначительные изменения освещения и т.д., а с другой, она учитывает изменения, которые человек при просмотре в принципе видит, но они не несут никакой смысловой нагрузки, например мелкие движения или повороты объектов или камеры, изменения цветов в силу подстройки параметров камеры и другие. Для решения этой проблемы были проанализированы исследования, в которых решались схожие задачи и подходы, которые используются в них. Основная идея решения данной проблемы состоит в том, что если мы не хотим считать разницу между исходными кадрами, т.к. там много лишней информации, то нужно сделать некоторое преобразование с кадром, для того, чтобы попытаться оставить только нужную информацию и сравнивать уже её. После изучения литературы по данной теме было решено остановиться на трёх подходах. С одной стороны все выбранные подходы использовались в аналогичных или похожих задачах, где было показано, что работают они хорошо. С другой были выбраны принципиально разные преобразования, чтобы охватить наибольшее число подходов и сравнить, какой из них работает лучше.

В работах [11, 15] для подсчета Δ использовалась гистограмма изображения. Гистограмма изображения – это график распределения яркости пикселей данного изображения. Строится он следующим образом: все цветовое пространство разбивается на интервалы равной ширины, после чего считается относительное количество пикселей, попавших в каждый интервал. Пример построенной гистограммы изображения можно увидеть на рис. 7. В примере гистограммы построена по

(a) Исходный кадр



(b) Гистограмма изображения

Рис. 7: Пример гистограммы, построенной по изображению

черно-белому изображению, в случае цветного – гистограммы строится по каждому из RGB каналов в отдельности. Таким образом, разница между двумя кадрами F_k и F_{k+1} в данном случае вычисляется следующим образом:

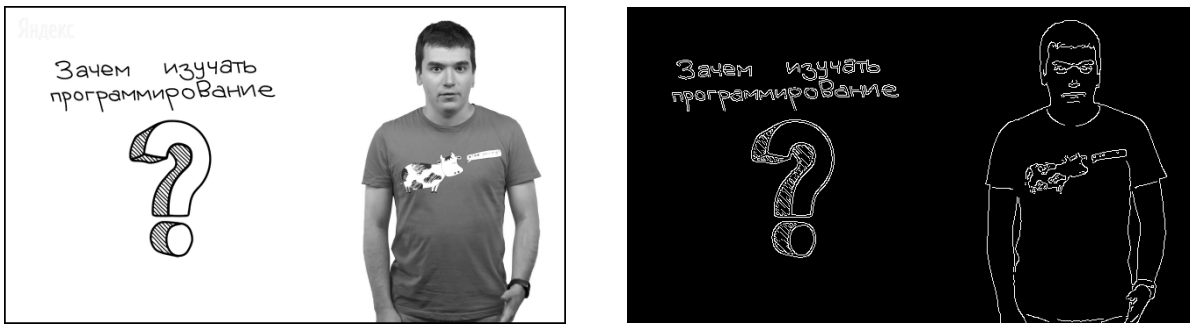
$$\Delta_k = \sum_{i=1}^{B}(Hist(F_k)[i] - Hist(F_{k+1})[i])^2 \tag{2}$$

Где $Hist(F)$ – гистограммы кадра F – вектор длины B, где на i-ом месте стоит количество пикселей, попавших в i-й интервал. B – количество интервалов, на которые было разбито цветовое пространство. Т.е. по сути это квадрат евклидового расстояния между двумя точками $Hist(F_k)$ и $Hist(F_{k+1})$ в пространстве $\mathbb{R}^B$.

В результате мы получили метрику, при подсчете которой учитываются только значения цветов пикселей, и никак не учитываются их положения. Это позволяет в меньшей степени учитывать шум, также этот метод нечувствителен к перемещению и повороту объектов в кадре. Но в то же время теряется очень много информации, что может негативно сказаться на результатах.

Следующим подходом, используемый также в работах [18, 26] был выбран детектор границ Кэнни. Результат данного преобразования представлен на рис. 8. Впервые этот метод был описан в работе [5]. Поиск границ происходит в несколько этапов:

1. Перед применением детектора происходит преобразование изображения к оттенкам серого, чтобы уменьшить вычислительные



(a) Исходный кадр (b) Найденные границы

Рис. 8: Пример применения к изображению детектора границ Кэнни

ресурсы.

2. Фильтрация шума, для этого обычно используют размытие изображения с помощью фильтра Гаусса. Ядро данного фильтра размера 5 с параметром распределения $\sigma = 1.4$ выглядит следующим образом:

$$\mathbf{K} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

3. Поиск приближенного значения градиента интенсивности, обычно с помощью оператора Собеля, применяя его в двух направлениях: горизонтальном (с ядром K_x, получая значение G_x) и вертикальном (с ядром K_y, получая значение G_y), после чего, вычисляется абсолютное значение (G) и угол (θ).

$$G = \sqrt{G_x^2 + G_y^2}, \quad \theta = arctan\left(\frac{G_y}{G_x}\right)$$

Затем значение угла округляется до одного из значений 0°, 45°, 90° либо 135°.

4. Подавление не максимумов. Для каждого пикселя в направлении градиента определяется, является ли значение градиента в данной точке локальным максимумом в данном направлении. Если

(a) Исходный кадр (b) После применения ДВП

Рис. 9: Пример применения к изображению дискретного вейвлет преобразования

пиксель является локальным максимумом, то значение его цвета устанавливается в максимально возможное, в противном случае – в минимальное.

5. Двойная пороговая фильтрация. Определяется два пороговых значения: высокое и низкое. Если значение градиента в точке больше, чем высокое пороговое – то данная точка попадает в выходное множество, определяющее границы на изображении. Если значение градиента в точке меньше низкого порогового значения, то она точно не попадает. Среди тех, что оказались между пороговыми значениями оставляются только те, что являются продолжениями границ, оказавшихся выше верхнего порогового значения.

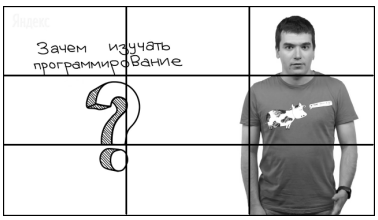
В результате получается бинаризованное изображения, причем такое, что количество пикселей, принимающее одно значение, значительно больше, чем другого. Таким образом сильно уменьшается количество информации, которое содержится в изображении. С другой стороны его по-прежнему легко интерпретировать, т.е. глядя на преобразованное изображения несложно понять, что на нем было изначально. Это говорит о том, что сильно возросла доля полезной информации, что должно хорошо сказаться на результатах работы алгоритма.

Последний подход, который использовался для решения аналогичных задач в работах [31, 16], это дискретное вейвлет преобразование (ДВП). Результат такого преобразования показан на рис. 9. Для од-

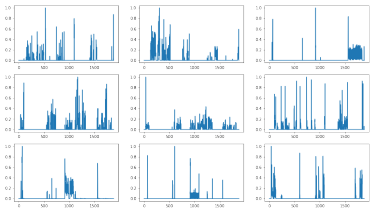
номерного сигнала простейшее ДВП строится следующим образом: все элементы последовательность группируются по 2 и для каждой пары считается полусумма и полуразность. На выходе получается 2 сигнала вдвое меньшей длины, один из которых аппроксимирует исходный сигнал, а второй содержит детализирующую информацию. Затем, к полусуммам применяется аналогичное преобразование и т.д. Двумерное ДВП – это композиция одномерных преобразований, примененных к строкам и столбцам матрицы исходного сигнала. После однократного применения ДВП к изображению на выходе получаем четыре изображения, одно из которых содержит уменьшенную размытую копию исходного, а три другие – это детализирующую информация. В работе [31] для сравнения изображений использовалась только детализирующая информация и было показано, что данный метод может успешно применяться для задачи извлечения ключевых кадров, такой же подход использовался и в данной работе. Таким образом, в каком-то смысле анализируется частотная информация изображения, делая акцент на высокочастотных составляющих, а анализируемые изображения похожи на результат работы детектора границ, с той разницей, что в данном случае изображение является не бинаризованным, а оттенками серого.

Вторая проблема наивного подхода заключается в том, что при подсчете разницы по формуле 1 происходит усреднение по всему кадру. В то время как на различных частях кадра может располагаться разный по сути и содержанию контент (например слева – слайды с текстом, справа – преподаватель, в углу – логотип организации и т.д.), который хорошо бы и анализировать отдельно, вычисляя различные метрики именно для данной части кадра. Таким образом в наивном подходе могут быть две различные ситуации, например переключение одного слайда с текстом на другой и один и тот же слайд, но с анимированным логотипом в углу кадра, в которых получится примерно одинаковое значение Δ, но по смыслу это абсолютно разные ситуации и одна из них значительно важнее для извлечения ключевых кадров, чем другая. Решения данной проблемы происходит в несколько шагов:

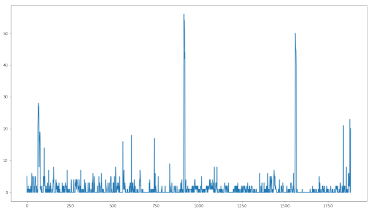
1. Разобьем кадр на K блоков размера $a \times b$ пикселей (рис. 10а).



(a) Разбиение на блоки



(b) Δ для каждого блока



(c) Δ для всего кадра

Рис. 10: Схема обработки кадра

Таким образом получим K различных видео, каждое длины такой же, как и исходное видео, но размер кадра в нем будет $a \times b$.

2. Посчитаем Δ для каждого блока отдельно и независимо от остальных(рис. 10b). После чего значения Δ для каждого блока бинаризуются, используя следующее пороговое значение:

$$Th^{i,j} = M^{i,j} + \alpha S^{i,j}$$

$$M^{i,j} = \frac{\sum_{k=1}^{N} \Delta_k^{i,j}}{N}$$

$$S^{i,j} = \sqrt{\frac{\sum_{k=1}^{N} (\Delta_k^{i,j} - M^{i,j})^2}{N}}$$

$Th^{i,j}$ – пороговое значение для блока (i, j), $\Delta_k^{i,j}$ – разница между k и $(k + 1)$ кадрами блока (i, j), α – константа, N – количество кадров в видео.

3. Вычислим Δ для всего кадра следующим образом:

$$\Delta_k = \sum_i \sum_j \Delta_k^{i,j} \tag{3}$$

и получим график изменений от кадра к кадру для исходного видео (рис. 10c). Т.е. теперь Δ_k – это количество блоков, в которых произошли значительные (по меркам каждого блока в отдельности) изменения, при переходе от k к $(k+1)$ кадру. А пики на графике Δ говорят о том, что произошли значительные изменения на большой части кадра.

Такой способ позволил уйти от абсолютных значений разницы к от-

носительным. И если в какой-либо части кадра большую часть времени происходят значительные изменения, то они не будут учтены в принципе, т.к. они не будут превышать $Th^{i,j}$, т.к. оно вычислено на основе средней разницы для данного блока.

Еще один плюс такого подхода в том, что имея информацию об относительных изменения в различных частях кадра, можно при подсчете Δ_k по формуле 3 задать различным $\Delta_k^{i,j}$ различные веса, в зависимости от их положения либо содержания. Например, т.к. наивный подход показал состоятельность эвристики с поиском людей и предположения о том, что человек в кадре – это преподаватель и, следовательно, информацию бесполезная для конспекта, то можно таким блокам, на которых есть люди, давать веса меньше, чем остальным. Или добавив предположение о том, что более важную информацию обычно стараются поместить ближе к центру – давать блокам, которые находятся ближе к центру большие веса.

Последней из больших проблем наивного алгоритма в том, каким образом происходит анализ посчитанных различий кадров. В наивном подходе Δ анализировалась для всего видео целиком, и это проблема похожа на предыдущую, т.к. в данном случае происходит усреднение по всему видео при поиске различных параметров для извлечения ключевых кадров. В то время как различные части видео могут быть очень разными по содержанию, например в начале лекции преподаватель показывает слайды с текстом, на которых возможно иногда делает пометки, а в конце видео он открывает текстовый редактор и начинает писать код. Контент в этих случаях абсолютно разный, соответственно нужны разные параметры при поиске ключевых кадров. Решение в данном случае также похоже на решение предыдущей проблемы. Если анализировать видео целиком неправильно, то нужно разбить видео на части. Наивный подход показал, что наличие людей в кадре – это важная информация, поэтому на основе ее и происходит разбиение видео на три типа интервалов, показанных на рис. 11. Далее, каждый интервалов анализируется отдельно и независимо от остальных. Такой подход позволил получить более однородные видео для анализа, что

(a) (b) (c)

Рис. 11: 3 типа интервалов видео: a – преподаватель отсутствует, b – преподаватель находится сбоку кадра, c – преподаватель находится по центру и занимает большую часть кадра

в целом хорошо сказалось на качестве извлечения ключевых кадров, но с другой стороны добавил необходимость дополнительно анализировать границы интервалов, чтобы понять, нужно ли добавлять их во множество ключевых кадров. Для этого нужно рассмотреть некоторое количество частных случаев, в зависимости от того, границы каких интервалов анализируются, в каком порядке они расположены и что на них изображено.

Общая схема работы алгоритма извлечения ключевых кадров представлена в приложении А, из которой видно, что за обработкой границ интервалов следует еще один этап – постобработка. На данном этапе происходит две вещи. Первая – это удаление повторов, т.к. может случиться такая ситуация, что ключевые кадры, извлеченные из соседних интервалов оказались одинаковыми. Вторая – уменьшение количества ключевые кадров. Если по какой-либо причине алгоритм выдал слишком много таких кадров, то из них выбираются наиболее значимые. Значимость определяется по величине пика на графике Δ, соответствующему данному ключевому кадру и продолжительности участка видео, который представляет данный ключевой кадр, т.е. если какая-либо информация долго находилась в кадре, значит преподаватель считает ее важной, поэтому вероятность попадания такого кадра во множество ключевых выше, чем у других кадров.

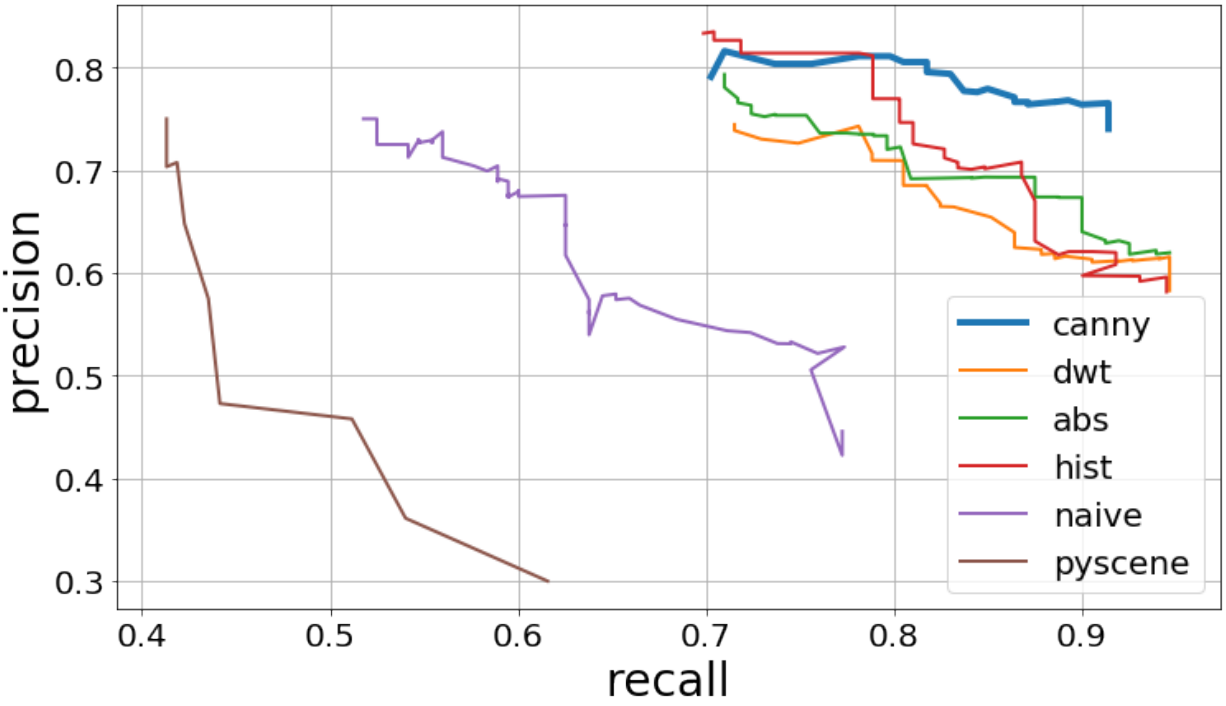


Рис. 12: Сравнение различных реализаций извлечения ключевых кадров

2.4. Сравнение

После реализации алгоритма, описанного в предыдущем разделе, было произведено сравнение качества его работы с наивным подходом и с существующим инструментом в виде PySceneDetect. Для проведения сравнения было отобрано несколько видеолекций, общей продолжительностью около двух часов. Для каждого видео была получена экспертная оценка в виде набора непересекающихся интервалов I, таких, что идеальный конспект – это ровно по одному кадру из каждого интервала. Затем, после того, как некоторая реализация выдавала набор ключевых кадров F, вычислялись две метрики: полнота($recall$) и точность($precision$). Для этого сначала определялось множество $T \subset F$ полезных кадров, это такие кадры, каждый из которых с одной стороны содержится в интервале из экспертной оценки, с другой, среди них нет двух кадров, попадающих в один и тот же интервал. Тогда $recall$ и $precision$ определяются следующим образом:

$$recall = \frac{|T|}{|I|}, \quad precision = \frac{|T|}{|F|}$$

Таким образом и полнота и точность лежат в интервале $[0, 1]$ и чем они больше, тем выше качество извлечения ключевых кадров алгоритмом. Затем, каждый алгоритм был проверен на размеченных данных по несколько раз, с разными параметрами.

Результаты сравнения представлены на рисунке 12. Из них видно, что наивным подход (*naive* на графике) работает значительно лучше, чем PySceneDetect (*pyscene* на графике). А описанный в предыдущем разделе алгоритм работает значительно лучше, чем наивный подход. Причем были протестированы различные его реализации, в зависимости от того, каким образом считалась разница между блоками соседних кадров. Лучше остальных показал себя подход с детектором границ Кэнни (*canny* на графике), остальные реализации работают примерно одинаково.

3. Распознавание аудио

3.1. Обзор инструментов для распознавания аудио

Для решения задачи преобразования аудио в текст было решено воспользоваться одним из существующих решений, т.к. с одной стороны создание своего инструмента не представляется возможным в рамках данной работы, а с другой не имеет смысла, т.к. над данной проблемой работают достаточно продолжительное время и есть качественные результаты в данной области.

Среди существующих инструментов есть два основных типа. Первый – это инструменты, взаимодействие с которыми происходит через HTTP API, для их работы необходимо подключение к сети Интернет и их использование либо платно, либо имеет некоторые ограничения на количество запросов и длину аудио в этих запросах. Данные инструменты максимально просты в использовании, имеют достаточно мало настроек, которые могут влиять на результат распознавания, и при их использовании пользователи практически ничего не знают про то, как на самом деле они работают и взаимодействие с ними происходит в формате "черного ящика".

Вторым типом инструментов по распознаванию речи являются фреймворки, которые позволяют по определенным правилам построить свою собственную систему распознавания речи. Примером такого инструмента является CMU Sphinx [27]. Для построения такой системы нужно задать несколько вещей, из которых обычно выделяют следующие: акустическая модель – отвечает за сопоставление фрагментов аудио фонеме, словарь – сопоставление словам их транскрипции, грамматика – правила построения словосочетаний и предложений, и языковая модель – это описание вероятности слов и словосочетаний в языке.

Преимуществами инструментов первого типа являются простота использования и обычно более высокая точность, ввиду того, что у таких систем есть доступ к большому количеству данных и вычислительных ресурсов. Второй тип инструментов обычно используется если нет до-

ступа к сети Интернет, либо в случаях, когда количество фраз, которые нужно будет распознавать заранее известно и их число не очень большое. Например, системы голосового управления каким-либо прибором. В этом случае можно построить соответствующие модели, которые описывают именно данный случай и получить более высокую точность, чем у инструментов первого типа.

Так как в данной работе заранее не известен набор фраз, который нужно будет распознавать и доступ в Интернет есть, то было решено остановиться на инструментах первого типа.

3.2. Реализация распознавания аудио

Большинство курсов на платформе stepik.org на русском языке, поэтому возможность работать с данным языком являлось ключевым фактором при выборе инструмента по распознаванию речи. С учетом возможности бесплатного использования практически единственным вариантом является сервис YandexSpeechKit [36]. При бесплатном использования сервис имеет следующие ограничения: 1000 запросов в сутки и 20 секунд аудио на запрос. Первое ограничение не вносит никаких дополнительных сложностей, так как при условии второго ограничения получается, что есть возможность распознавать больше пяти часов аудио в сутки, что значительно больше, чем скорость появления новых видео на stepik.org. Второе ограничения вносит дополнительные сложности, так как обычно видеолекция длится больше 20 секунд, это значит, что для того, чтобы преобразовать ее в текст, нужно разбивать ее на интервалы по 20 секунд или меньше, и распознавать их отдельно. Проблема тут заключается в том, что если при разбиении часть слова попала в один интервал, а часть в другой, то данное слово заведомо будет распознано неправильно, т.к. обрабатываются запросы независимо друг от друга. Для решения данной проблемы было сделано 2 предположения:

- Слово произносится меньше 5 секунд
- Между словами наблюдается локальный минимум громкости

Затем разбиение происходило следующим образом: на интервале от 15 до 20 секунды находится минимум громкости, который исходя из вышеописанных предположений является паузой между словами, и по этому минимуму происходит разбиение. Оставшаяся часть аудио обрабатывается аналогичным образом. Экспертная оценка показала, что в подавляющем большинстве случаев предположения соответствуют действительности и разбиение действительно происходит между словами.

4. Публикация конспекта

4.1. Требования к инструменту для публикации

Перед выбором инструмента для предоставления пользователям доступа к сгенерированному конспекту, был сформулирован следующий набор требований к такому инструменту:

- Бесплатность.
- Совместное редактирование. Как было сказано во введении, конспект в виде ключевых кадров и полного текста лекции не в полной мере решает описанные проблемы видеокурсов, поэтому было решено дать пользователям возможность вносить правки в сгенерированный конспект. Это с одной стороны уменьшит сложность создания конспекта, т.к. есть предварительная версия, а с другой позволит получить конспект более качественный, чем тот, который бы создавался одним человеком.
- Интерфейс разрешения конфликтов. Т.к. возможное количество слушателей курса не ограничено, и в принципе онлайн-обучение рассчитано на большие массы людей, то нужно уметь разрешать ситуации, когда несколько пользователей внесли в один и тот же конспект различные изменения.
- История изменений. Т.к. редактировать конспекты в итоге может любой желающий, то нужна защита от намеренной либо случайной порчи созданного конспекта. В данном случае для этого используется история изменений, которая позволяется откатиться к любому предыдущему состоянию.
- Система прав доступа. Это нужно, чтобы была возможность выделить некоторую группу людей, например преподавателей курса или кого-то из команды stepik.org, которым позволено больше, чем остальным, чтобы иметь возможность решать конфликтные ситуации в ходе создания конспекта.

- Поддержка LaTeX. Значительная часть курсов на stepik.org имеет техническую направленность, и для создания хорошего конспекта по данным курсам зачастую нужна возможность использовать специальные символы, математические формулы и т.д.

- Экспорт в pdf. У пользователей должная быть возможность скачать необходимые конспекты в удобном для чтения формате, чтобы получить к ним доступ в любое время.

- Поддержка сложной структуры страниц. В первой главе данной работы была описана структура сущностей на stepik.org и конспекты должны повторять эту структуру, чтобы облегчить пользователям поиск нужного конспекта.

4.2. Выбор инструмента для публикации

Под эти требования подходят различные wiki. Ресурс wikimatrix [34] предоставляет удобный инструмент для поиска и сравнения различных wiki. После изучения предоставленной им информации было решено выбрать в качестве сервиса для публикации конспектов MediWiki [19]. Это wiki с открытым исходным кодом, богатой системой плагинов, за счет которых достигается функциональность, описанная в требованиях выше. Также этот инструмент имеет большое сообщество и используется многими популярными сайтами, в том числе википедией. В качестве расширений были установлены WikiEditor, OpenID, CategoryTree, Math, PdfBook и некоторые другие.

5. Сервис по генерации конспектов

Сервис по генерации конспектов предоставляет собой отдельный от stepik.org web сервер. Взаимодействие с пользователей происходит через HTTP API. Поддерживается только один вид запроса от пользователя, в котором он должен указать тип сущности, для которой требуется создать конспект (курс, модуль, урок или шаг) и уникальный идентификатор этой сущности на stepik.org. После того, как пользователь делает запрос сервис последовательно выполняет следующие операции:

- Валидация данных. Происходит проверка введенных пользователем значений на корректность. Если они некорретны, то работа завершается, о чем сообщается пользователю.

- Создание задачи и отправка ее на исполнение в пул процессов. Таким образом достигается параллельность генерации конспектов для различных сущностей.

- Получение данных со stepik.org. Используя публичный Stepik API, сервис получает все необходимые для генерации конспекта данные.

- Извлечение ключевых кадров. Используя алгоритм, описанный во второй главе данной работы происходит извлечение ключевых кадров из всех видеолекций, полученных на предыдущем этапе.

- Публикация изображений. Для хранения изображений используется сторонний хостинг uploadcare [32], хотя mediawiki позволяет хранить изображения, это сделано для того, чтобы уменьшить зависимость от различных частей сервиса, чтобы в будущем, при необходимости было проще заменить инструмент для публикации конспектов на другой.

- Распознавание аудио. Из видеолекции извлекается аудиоряд, разбивается на интервалы по описанному в главе 3 принципу и, используя YandexSpeechKit, распознается в текст.

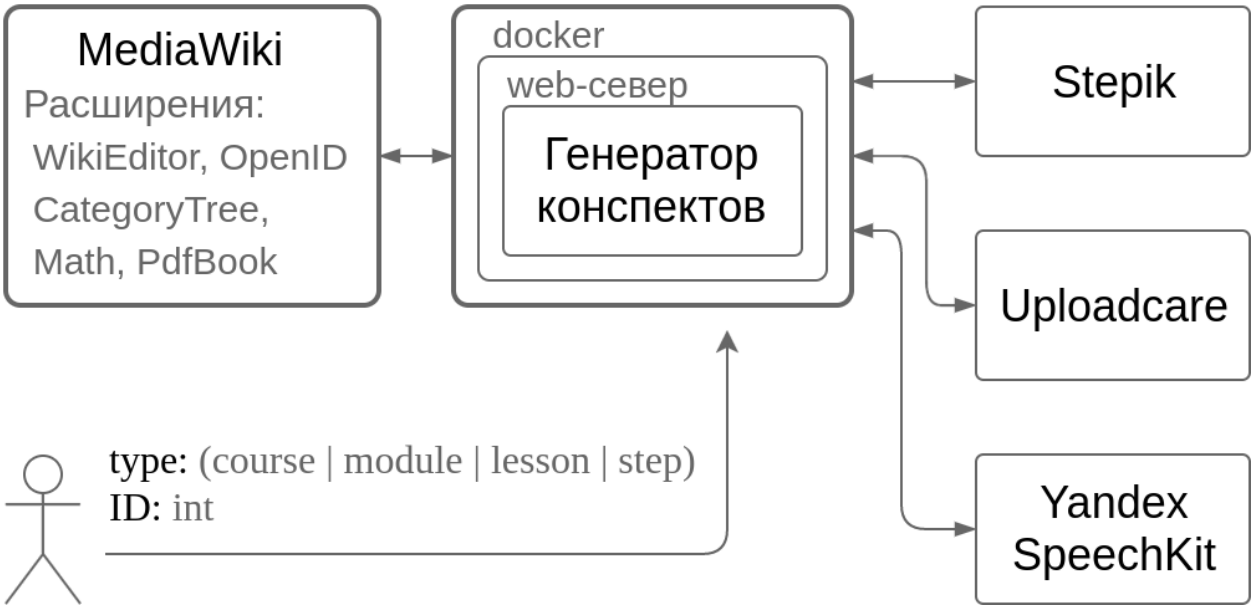


Рис. 13: Схема работы сервиса по генерации конспекта

- Сборка конспекта. Из ссылок на изображения, соответствующие ключевым кадрам, и набор текстов, соответствующим интервалам аудио, собирается единый документ, в формате, необходимым для публикации на mediawiki.

- Публикация конспекта. Полученный на предыдущем шаге документ публикуется на отдельном сервере с mediawiki, используя mediawiki API.

Для удобства разворачивания сервиса он оформлен в виде docker-образа, это избавляет от зависимости от окружения и необходимости устанавливать множество библиотек и инструментов, которые используются сервисом для генерации конспекта, а также упрощает дальнейшую разработку сервиса. Общая схема работы сервиса по генерации конспекта представлена на рисунке 13.

Заключение

В рамках данной работы были достигнуты следующие результаты:

- Изучено и проанализировано большое количество актуальных исследований по теме извлечения ключевых кадров и смежным задачам.
- Разработан и реализован алгоритм по извлечению ключевых кадров для задачи генерации конспекта по видеокурсу, который, по результатам проведенного сравнения, работает лучше, чем существующие решения.
- В ходе решения поставленных задач была освоена работа со множеством различных инструментов и сервисов, среди которых Stepik, YandexSpeechKit, Uploadcare, MediaWiki, Docker, OpenCV, dlib и др.
- Реализован сервис по генерации конспектов, который инкапсулирует в себе всю логику по созданию конспектов, предоставляя пользователю максимально простой и удобный интерфейс.

В качестве дальнейших направлений работы можно выделить:

- Запуск wiki с конспектами на широкую аудиторию студентов для получения обратной связи от непосредственных пользователей системы.
- Расширение множества видео, на которых может работать алгоритм извлечения ключевых кадров.
- Применения методов обработки естественного языка для обработки сгенерированного конспекта с целью уменьшить количество правок, необходимое для приведения конспекта к конечному варианту.

Список литературы

[1] Albanie. Сравнение различный реализаций для извлечения ключевых кадров. — 2017. — Режим доступа: `https://github.com/albanie/shot-detection-benchmarks`.

[2] Amiri Ali, Fathy Mahmood. Hierarchical keyframe-based video summarization using QR-decomposition and modified k-means clustering // EURASIP Journal on Advances in Signal Processing. — 2010. — Vol. 2010. — P. 102.

[3] Balasubramani R, Kannan Dr V. Efficient use of MPEG-7 color layout and edge histogram descriptors in CBIR systems // Global Journal of Computer Science and Technology. — 2009. — Vol. 9, no. 4.

[4] Calandra Brendan, Brantley-Dias Laurie, Dias Michael. Using digital video for professional development in urban schools: A preservice teacher's experience with reflection // Journal of Computing in Teacher Education. — 2006. — Vol. 22, no. 4. — P. 137–145.

[5] Canny John. A computational approach to edge detection // IEEE Transactions on pattern analysis and machine intelligence. — 1986. — no. 6. — P. 679–698.

[6] Cernekova Zuzana, Pitas Ioannis, Nikou Christophoros. Information theory-based shot cut/fade detection and video summarization // IEEE Transactions on circuits and systems for video technology. — 2006. — Vol. 16, no. 1. — P. 82–91.

[7] Chang Wen-Hsuan, Wu Yu-Chieh, Yang Jie-Chi. Webpage-based and video summarization-based learning platform for online multimedia learning // International Conference on Technologies for E-Learning and Digital Entertainment / Springer. — 2011. — P. 355–362.

[8] Chang Wen-Hsuan, Yang Jie-Chi, Wu Yu-Chieh. A keyword-based video summarization learning platform with multimodal surrogates //

Proceedings of the 2011 IEEE 11th International Conference on Advanced Learning Technologies / IEEE. — 2011. — P. 37–41.

[9] Comparison of text and video cases in a postgraduate problem-based learning format / Thomas Balslev, Willem S De Grave, Arno MM Muijtjens, AJJA Scherpbier // Medical education. — 2005. — Vol. 39, no. 11. — P. 1086–1092.

[10] Coursera. Платформа онлайн-образования. — 2017. — Режим доступа: `https://www.coursera.org/`.

[11] Divakaran Ajay, Peker Kadir A, Sun Huifang. Video summarization using motion descriptors // Photonics West 2001-Electronic Imaging / International Society for Optics and Photonics. — 2001. — P. 517–522.

[12] Ekin Ahmet, Tekalp A Murat, Mehrotra Rajiv. Automatic soccer video analysis and summarization // IEEE Transactions on Image processing. — 2003. — Vol. 12, no. 7. — P. 796–807.

[13] Farin Dirk, Effelsberg Wolfgang, de With Peter HN. Robust clustering-based video-summarization with integration of domain-knowledge // Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on / IEEE. — Vol. 1. — 2002. — P. 89–92.

[14] Ffprobe. — 2017. — Режим доступа: `https://ffmpeg.org/ffprobe.html`.

[15] Fujimura Kenichi, Honda Koichiro, Uehara Kuniaki. Automatic video summarization by using color and utterance information // Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on / IEEE. — Vol. 1. — 2002. — P. 49–52.

[16] Gianluigi Ciocca, Raimondo Schettini. An innovative algorithm for key frame extraction in video summarization // Journal of Real-Time Image Processing. — 2006. — Vol. 1, no. 1. — P. 69–88.

[17] Gong Yihong, Liu Xin. Video summarization and retrieval using singular value decomposition // Multimedia Systems. — 2003. — Vol. 9, no. 2. — P. 157–168.

[18] Khurana Khushboo, Chandak MB. Key frame extraction methodology for video annotation // International Journal of Computer Engineering and Technology. — 2013. — Vol. 4, no. 2. — P. 221–228.

[19] MediaWiki. Wiki engine. — 2017. — Режим доступа: `https://www.mediawiki.org/`.

[20] Mundur Padmavathi, Rao Yong, Yesha Yelena. Keyframe-based video summarization using Delaunay clustering // International Journal on Digital Libraries. — 2006. — Vol. 6, no. 2. — P. 219–232.

[21] PySceneDetect. — 2017. — Режим доступа: `http://pyscenedetect.readthedocs.io/`.

[22] Rahmat Roushanak, Malik Aamir Saeed, Kamel Nidal. Comparison of LULU and median filter for image denoising // International Journal of Computer and Electrical Engineering. — 2013. — Vol. 5, no. 6. — P. 568.

[23] Santagata Rossella. Designing video-based professional development for mathematics teachers in low-performing schools // Journal of teacher education. — 2009. — Vol. 60, no. 1. — P. 38–51.

[24] Sappa Angel, Dornaika Fadi. An edge-based approach to motion detection // Computational Science–ICCS 2006. — 2006. — P. 563–570.

[25] Shotdetect. — 2017. — Режим доступа: `http://johmathe.name/shotdetect.html`.

[26] Sonara Abhilash K, Brahmbhatt Pinky J, Chaudhary Manoj D. Applying edge density based region growing with frame difference for detecting moving objects in video surveillance systems // International Journal of Research in Engineering and Technology (IJRET). — 2014. — Vol. 3, no. 04. — P. 693–699.

[27] Sphinx CMU. Фреймворк для распознавания речи. — 2017. — Режим доступа: https://cmusphinx.github.io/.

[28] Stepik.org. Платформа онлайн-образования. — 2017. — Режим доступа: https://stepik.org/.

[29] Summarization of videotaped presentations: automatic analysis of motion and gesture / S.X. Ju, M.J. Black, S. Minneman, D. Kimber // IEEE Transactions on Circuits and Systems for Video Technology. — 1998. — Vol. 8, no. 5. — P. 686–696.

[30] Swain Michael J. Interactive indexing into image databases // IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology / International Society for Optics and Photonics. — 1993. — P. 95–103.

[31] Tint Khin Thandar, Soe Dr. Kyi. Key Frame Extraction for Video Summarization Using DWT Wavelet Statistics // International Journal of Advanced Research in Computer Engineering & Technology. — 2013. — may. — Vol. 2, no. 5. — P. 1829–1833.

[32] Uploadcare. Uploadcare – file uploads, processing, storage, and delivery for web and mobile apps. — 2017. — Access mode: https://uploadcare.com/.

[33] Video summarization: techniques and classification / Muhammad Ajmal, Muhammad Ashraf, Muhammad Shakir et al. // Computer Vision and Graphics. — 2012. — P. 1–13.

[34] WikiMatrix. Сравнение различных wiki. — 2017. — Режим доступа: http://www.wikimatrix.org/.

[35] Wolf Wayne. Key frame selection by motion analysis // Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on / IEEE. — Vol. 2. — 1996. — P. 1228–1231.

[36] YandexSpeechKit. Сервис разпознавания речи. — 2017. — Режим доступа: https://tech.yandex.ru/speechkit/cloud/.

[37] Yousef Ahmed Mohamed Fahmy, Chatti Mohamed Amine, Schroeder Ulrik. Video-based learning: a critical analysis of the research published in 2003-2013 and future visions. — 2014.

[38] Zhang Hongjiang, Low Chien Yong, Smoliar Stephen W. Video parsing and browsing using compressed data // Multimedia tools and applications. — 1995. — Vol. 1, no. 1. — P. 89–111.

Приложение А. Схема алгоритма извлечения ключевых кадров