Санкт–Петербургский государственный университет

Спиридонов Александр Юрьевич

Выпускная квалификационная работа

Локализация и распознавание внедрённых субтитров в видеопотоке

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2016 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Математическое и программное обеспечение вычислительных машин»

Научный руководитель:
Кандидат технических наук, доцент,
кафедра компьютерного моделирования
и многопроцессорных систем - Гришкин Валерий Михайлович

Рецензент:
Кандидат физико-математических наук, доцент кафедры компьютерных технологий и систем - Коровкин Максим Васильевич

Санкт-Петербург
2020 г.

Содержание

Введение

В связи с развитием информационных технологий, методы передачи и получения информации постоянно меняются от использования наиболее эффективных решений, как например текст, к решениям удобным для человеческого восприятия, но более затратным, с точки зрения передачи данных. Одним из возможных решений в наши дни является видео, в связи с массовым распространением и доступностью таких сервисов как YouTube, Twitch и т.п., ставшее одним из главнейших способов передачи информации. Однако видео материалы, как способ фиксирования тех или иных событий, рассчитаны на использование основных чувств человека, а именно зрение и слух. Для людей, которые, в данный момент, или вовсе не имеют возможности слушать звуковую составляющую видео, в наше время существуют субтитры. Текст субтитров является важной частью видеоматериала для понимания информации и её анализа. В данной работе рассматривается решение проблемы, появившейся вследствие существования субтитров, встроенных в видеопоток и соответственно, не имеющих текстовой версии, что не позволяет взаимодействовать с текстом в привычном варианте.

В данной работе автором поставлена задача создания алгоритма способного распознавать текст субтитров при любых даже наиболее проблемных видеорядах и возвращать его вместе с временными метками пользователю. Данную задачу можно разбить на несколько этапов: устранение шумов, обнаружение субтитров, определение кадров, содержащих текст, поиск моментов его смены и распознавание текста. Для решения данных задач автором были опробованы несколько методов, некоторые из которых показали себя недостаточно эффективными, как например, использование цветовых характеристик или гистограмм проекций. В то же время методы поиска контуров, статичных зон, метод “заливки” и связных компонент наоборот показали себя с лучшей стороны, что подробнее будет описано в работе.

Структура работы: постановка задачи, обзор литературы, глава 1 - предобработка кадров, глава 2 – анализ предобработанных кадров, реали-

зация и экспериментальные результаты с последующим выводом и заключение.

Алгоритм, описанный в данной работе, позволяет получать текст для его дальнейшего использования, например, при переводе субтитров, давая возможность задействовать машинный перевод. Помимо этого, при появлении алгоритма, устраняющего встроенные субтитры, появится возможность их замены включаемыми, если дополнить его результатами, которые мы планируем получить в конце исследования. Также существует возможность получения текста с последующим сжатием видео, но с сохранением качества текста субтитров. Извлечённые субтитры можно использовать как текст для анализа, который в совокупности с видео состоит в создании тематических меток для видео или определения эмоционального тона видео.

Постановка задачи

Проблема распознавания текста на изображении, в английской литературе называемая Optical Character Recognition (OCR) или оптическое распознавание символов, ставится как задача получения текста, присутствующего на изображении. При попытке применить тот же подход к видео файлу, мы встретимся со следующими трудностями: в отличии от обычного OCR не на каждом кадре видео присутствует текст, что вызывает необходимость разделения кадров на имеющие и не имеющие субтитры и отделения таковых от текста, существующего в видео на фоне. Другое немаловажное отличие от стандартной задачи OCR – временные рамки. Временные рамки – это границы времени, указывающие время начала и конца отображения строки субтитров. Их получение также является одной из задач, поставленных в данной работе. Выбрав кадры с текстом субтитров, мы переходим к задаче оптического распознавания символов, которая делится на два этапа: предобработка и распознавание. В основе данной работы лежит этап предобработки, так как при заранее неизвестном, неопределённом фоне он позволяет выделить искомый текст на изображении, без чего последующее распознавание невозможно или крайне затруднено. Цель предобработки — это повышение качества последующего распознавания, посредством максимально возможного “очищения” изображения или в нашем случае кадра

от всего, что не является необходимым для распознавания. Обычно предобработка для OCR состоит из следующих целей[10]:

- выравнивание текста до стандартного горизонтального;
- шумопонижение;
- бинаризация;
- удаление линий;
- анализ структуры, например, таблиц в строках и столбцах которых может находиться искомый текст;
- обнаружение текста;
- распознавание шрифта;
- локализация символов;
- нормализация размера и пропорции изображения.

Учитывая, что цель работы — это получение субтитров, то ввиду их природы, список принимает другой вид. Выравнивание текста в случае с субтитрами не является необходимым, так как они изначально текст и соответственно горизонтальны. Удаление линий не является конкретной задачей, а является частью задачи по отделению субтитров от фона, которому принадлежат горизонтальные линии. В вышеуказанном списке целей она обозначена как шумопонижение. Отсутствие структуры видео устраняет необходимость её анализа. Распознавание шрифта необходимо при смене такового в тексте для подбора соответствующего алгоритма. Однако для удобства восприятия зрителя, шрифт редко меняется по середине видео. При этом шрифт определяется с помощью стандартов [2], соответственно, проблема распознавания шрифта исчезает. В итоге, учитывая, что изображения одинаковые, в виду природы кадров, устраняя нормализацию размера и пропорций изображений, список целей предобработки можно представить следующим образом:

- шумопонижение;
- бинаризация;
- обнаружение текста;
- локализация символов.

Последним, но не менее важным этапом является распознавание текста. Распознавание текста может быть сложной задачей в зависимости от работы, проделанной на этапе предобработки, цель которой упростить данный этап. Дополнительным фактором, упрощающим решения данной задачи, является правильный подбор алгоритма, учитывающего особенности искомого текста.

Обзор существующих методов получения субтитров встроенных в видеопоток

Проблему получения субтитров, встроенных в видеопоток уже пытались решать при помощи выделения границ текста субтитров[16], алгоритм показал неплохую точность. Недостатками работы алгоритма являются ограничения на положение субтитров снизу, а также отсутствует способ классификации кадров, имеющих в наличии субтитры. Вместо этого, в данном методе, был выбран сегмент видео на всём промежутке которого присутствуют субтитры. Также у алгоритма имеются проблемы при работе с видео, содержащим контрастные элементы, что подробнее будет описано автором дальше.

В работе Zarifar B. [17], которая представляет алгоритм для локализации и классификации субтитров в ТВ видео, была попытка показывающая неплохие результаты[17], но она непосредственно использует особенности телевизионного сигнала и приставки его расшифровывающей для своей работы и поэтому не подходит для работы с обычным видео

При решении данной проблемы имело место и использование нейронных сетей[15], которое не имеет ограничений в подходе выделения границ[16]. Существующий алгоритм, как и предыдущий, использовал информацию о

соседних кадрах для повышения качества распознавания. В конечном счёте, алгоритм показал неплохие результаты в 86% точности распознавания текста, но кадры не получали никакой предобработки, что позволяет сделать предположение о возможном улучшении результатов.

При рассмотрении существующих программных решений программных решений, было обнаружено 4 существующих решения: SubRip[12], CCExtractor[4], Burnt-in subtitle extractor[3] и videocr[14]. Большинство данных решений за исключением SubRip используют Tesseract для распознавания текста с кадров видео. SubRip, в свою очередь, требует точной настройки человеком, который выберет параметры предобработки, укажет локацию субтитров, а также исправит ошибки посимвольного распознавания. Следовательно, алгоритм нельзя назвать автоматическим, за счёт человека возможно достижение высокого качества результатов. CCExtractor и Burnt-in subtitle extractor уже более автоматизированные решения, которым тем не менее, тоже необходим человек для указания цвета субтитров для предобработки в обоих программах и цветов контуров в Burnt-in subtitle extractor. Последнее решение videocr не использует предобработку, а полагается полностью на Tesseract с последующим анализом результатов. Положительное отличие от остальных является максимально автономным с опциональными параметрами настройки распознавания.

Глава 1. Предобработка кадров

1.1 Выделение границ

Одной из целей автора было создание универсального метода, работающего с любым видеорядом на фоне субтитров. Для достижения этой цели все методы проверялись на самом проблемном для них типе видео - контрастной, мало изменяющейся анимации.

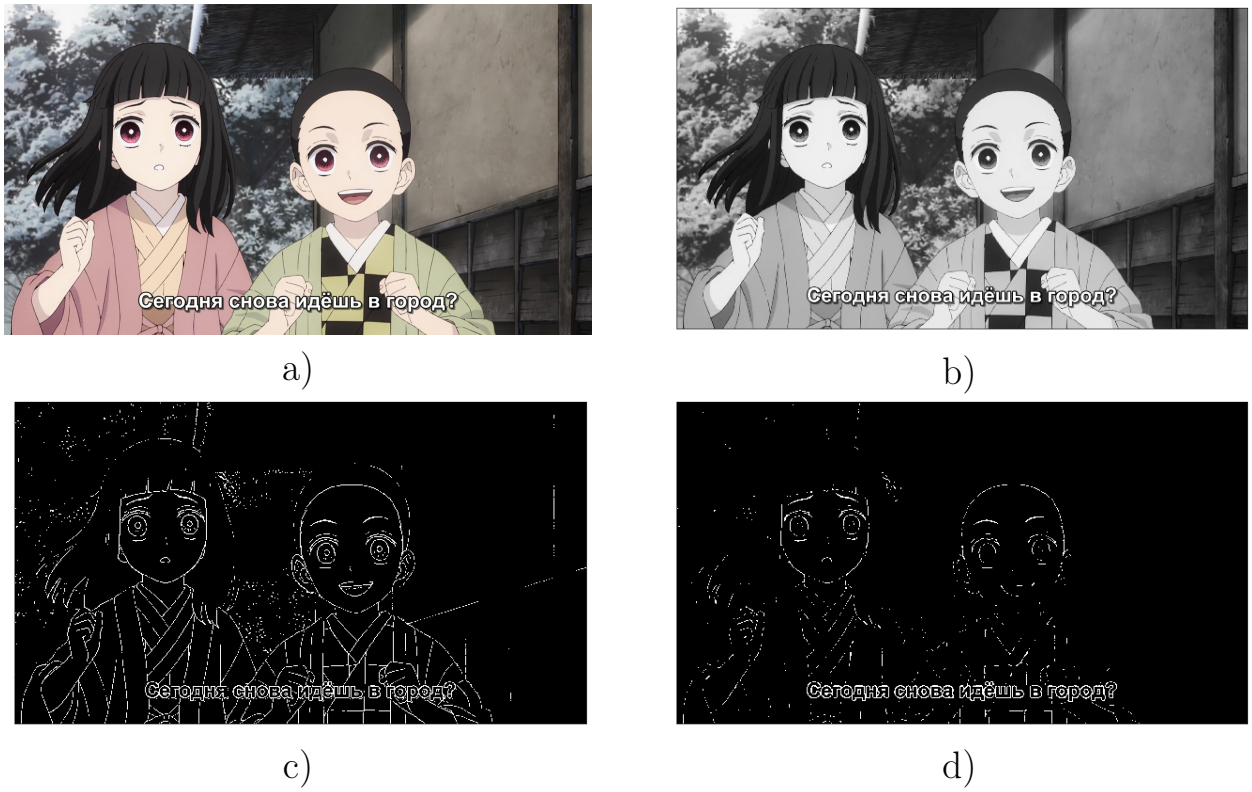
Аналогичный таковому из [16] этот метод пытается найти субтитры за счёт их контрастности. Приводим кадры видео к чёрно-белому формату, так как это необходимо для работы последующих методов обработки изображений рис.1b, что в последующем делает алгоритм независимым от цвета текста субтитров. Поиск границ текста осуществляется методом adaptive thresholding из библиотеки работы с изображениями OpenCV[9]. Суть метода в поиске выделяющихся точек на фоне за счёт подсчёта свёртки окна выбранного размера с искомой точкой, а точнее пикселем в центре и сравнением результата для построения бинарной карты особых точек, как в формуле 1.

$$dst(x,y) = \begin{cases} 0, & if\ src(x,y) > T(x,y) \\ maxValue,\ Otherwise \end{cases} \tag{1}$$

Где src(x,y) это искомое изображение, а T(x,y) это функция свёртки, которой в моём случае была функция Гаусса[6]. По экспериментам, проведённым с предварительным размытием чёрно-белого изображения при помощи размытия Гаусса[6], такая обработка черно-белого изображение перед применением adaptive thresholding повышает качество бинарного изображения убирая высокочастотный шум рис.1.

1.2 Гистограммы

После получения бинарного изображения краевых точек, сначала было проверено получится ли найти субтитры только по этим результатам. Аналогично [16] спроецировав точки бинарного изображения из предыдущей части на вертикальные и горизонтальные стороны изображения. Та-



a) b)

c) d)

Рис. 1: Кадр во время обработки: a) изначальное изображение, b) приведённое к чёрно белому, c) контуры без размытия, d) контуры с размытием.

ким образом получены гистограммы горизонтальной и вертикальной оси изображения, на которых найдя резкое изменение нашлись бы соответственно края субтитров.

а) б)

Рис. 2: Гистограммы проекций кадра с низкой контрастностью: а) проекция на вертикальную ось б) проекция на горизонтальную ось.

Данный метод хорошо себя показал на субтитрах на фоне видео ряда, не отличающегося контрастностью, как например кинофильм с реальными сценами и актёрами, см. рис.2. Но когда дело доходит до высококонтрастных изображений, как например мультипликация и анимация, появляют-

ся проблемы с появлением шумов вызванных той самой контрастностью. Соответственно при большой зашумлённости становиться проблематично нахождение краёв субтитров, см. рис.3.

а) б)

Рис. 3: Гистограммы проекций кадра с высокой контрастностью: а) проекция на вертикальную ось б) проекция на горизонтальную ось.

1.3 Статичные регионы

Данный метод уже использовался в [7, 17] и в немного изменённом виде в [15]. Суть метода состоит в использовании временного свойства субтитров, а именно их присутствие на нескольких последовательных кадрах. Найдя общую часть этих кадров можно найти сами субтитры, убрать посторонний шум и что немаловажно отделить субтитры от текста на видео. Если в [17] метод использовался параллельно с обнаружением краёв, после чего их результаты объединялись,было решено для повышения качества выделения границ 1.1 объединить их вместе, ища общие части бинаризованных изображений из параграфа "Выделение границ".

Данная модификация сильно повысила качество бинаризации устранив множество шумов. Тем не менее некоторые шумы всё ещё остались. Одно из возможных решений данной проблемы это увлечение количества кадров для поиска статических регионов. После данной модификации исчезают почти все оставшиеся шумы, но ценой за это является растущая сложность метода вследствие сложности подсчёта временных границ субтитров, см. рис.4. В конечном алгоритме данной работы используется описанный выше вариант объединения двух последовательных кадров с последующим применением описанных далее методов.

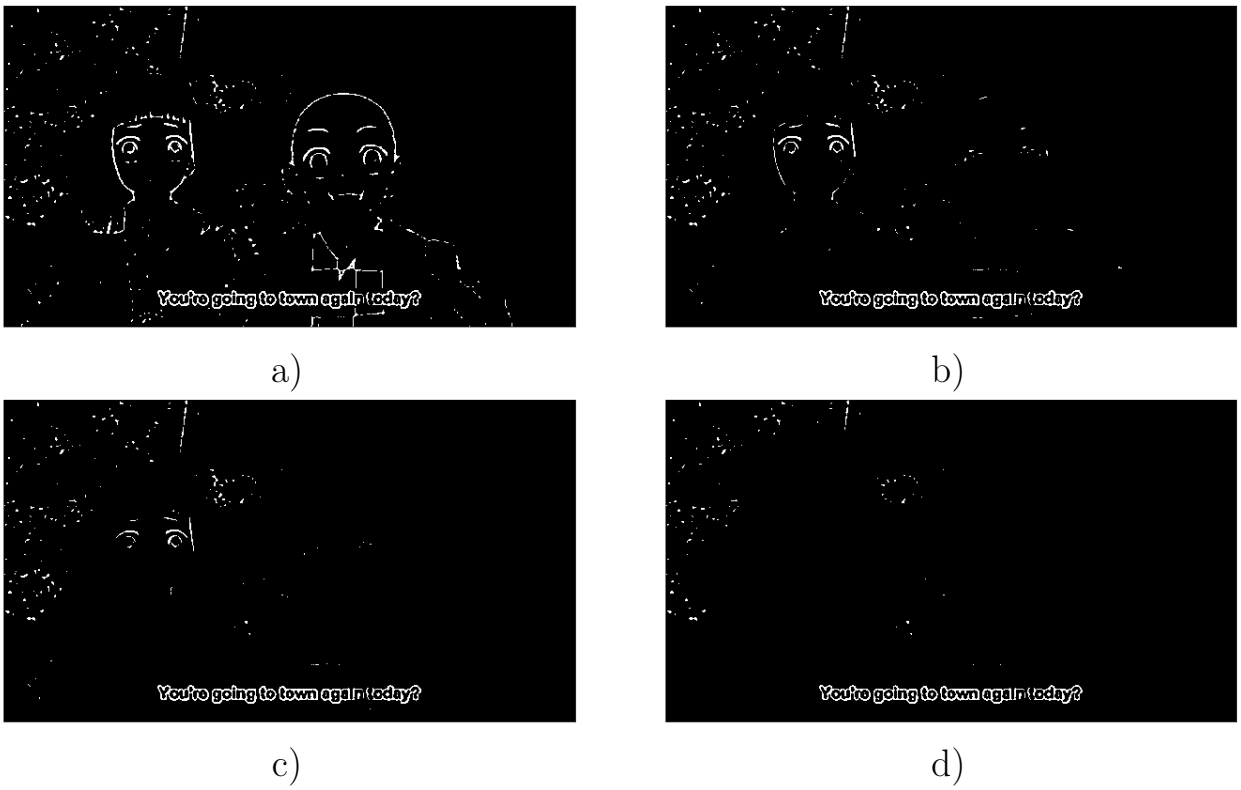


Рис. 4: Статичные области: a) изначальное изображение, b) 2 кадров, c) 5 кадров, d) 10 кадров.

1.4 Метод Заливки

Результатом всех предыдущих операций на первый взгляд кажется кадр с текстом, который можно использовать в задаче распознавания. Однако при попытке распознавания обнаруживается проблема, не бросающаяся в глаза в виду нашего огромного опыта чтения, но влияющая на алгоритмы. После вышеописанной предобработки у нас не обычный текст, но его контуры, что отличается от обычной задачи OCR. Проба распознавания такого изображения с использованием Tesseract показала неудовлетворительные результаты.

Для решения данной проблемы были найдены два пути: создать или обучить алгоритм распознавания для контуров или придумать способ преобразования контуров в классические буквы. Сначала было решено попробовать второй способ так как это, относительно создания алгоритма распознавания, занимает мало времени и сразу будут видны примерные результаты.

Придуманное решение состояло в последовательном применении мор-

фологической операции расширения и перекрашивании изображения в белое при помощи “заливки”. Морфологическая операция расширения состоит в прохождении по изображению заданного окна – ядра, в котором центральный пиксель заменяется максимальным значением в окне. Данная операция по сути расширяет или иначе утолщает все белые точки изображения что необходимо для создания замкнутых областей внутри контуров букв, чтобы они остались чёрными на изображении после заливки. Заливка состоит в выборе чёрного пикселя и перекрашивания всех касающихся его чёрных одним цветом, который в нашем случае белый, см. рисунок 5. В коде данный метод выполнен как последовательное применение к изображению операций dilate с ядром три на три из единиц и floodFill с нулевой маской и максимальным значением в 255, взятые из OpenCV.

Рис. 5: Результат работы заливки

Несмотря на то что данный метод создавался изначально просто чтобы превратить контуры в полноценные буквы, его применение показало максимальную эффективность в задаче устранения шумов в виду их объединения с закрашенной областью. Также полученное изображение является аналогичным таковому из наиболее изученного и старого вида задачи распознавания символов – чёрный текст на белом фоне. Данный результат не только исключает необходимость в создании специализированного алгоритма распознавания, но использовать стандартные предобученные алгоритмы и как будет показано в следующей главе использовать помимо алгоритмов, рассчитанных на целый текст также и алгоритмы распознающие символы по отдельности.

Глава 2. Анализ предобработанных кадров

2.1 Связные компоненты

Для определения местонахождения субтитров среди найденных точек, было решено использовать метод связных компонент, который находит связные области на изображении, возвращая их положение и характеристики. Используя эти данные можно отсортировать полученные области по размеру отсеяв слишком малые для текста и получив границы местонахождения субтитров, см.рис.6, что было выполнено при помощи функции connectedComponentsWithStats из OpenCV давшей необходимые данные.

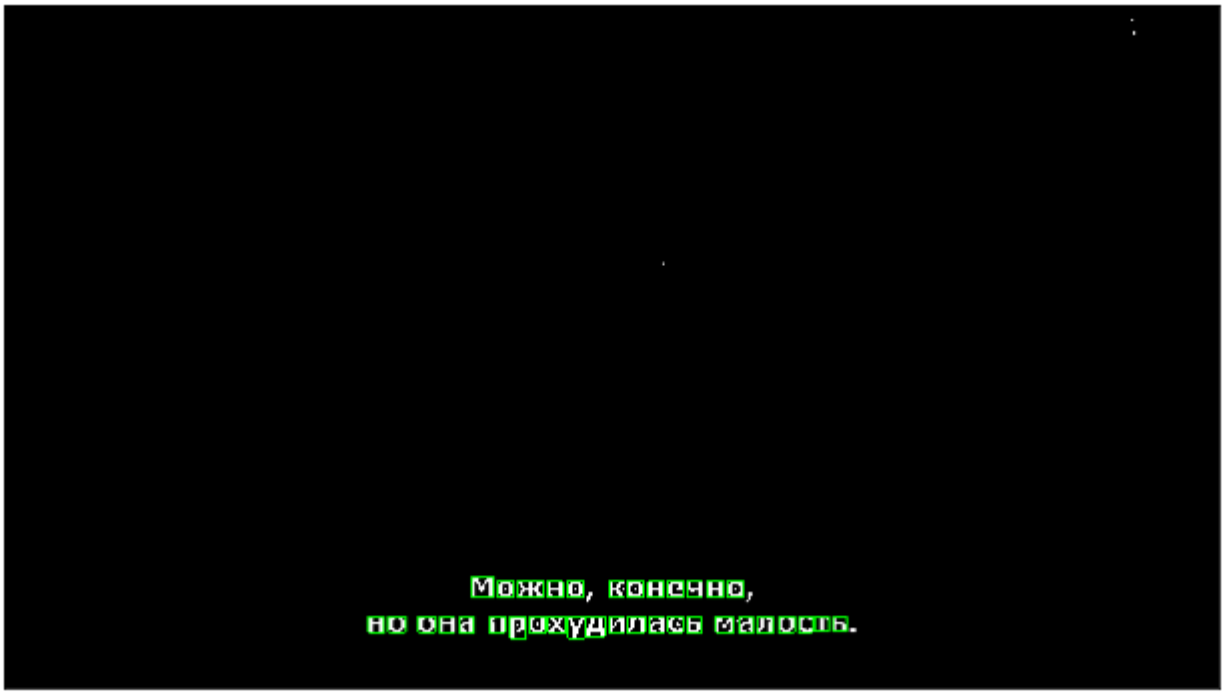


Рис. 6: Результат поиска связных областей.

Рассмотрев позицию связных областей можно, сделать вывод о том содержатся ли в этом кадре субтитры, так как если это утверждение верно, то проявится характерное свойство текста, а именно горизонтальность. Посчитав дисперсию вертикальных позиций связных компонент,она проверяется на превышение порога. Если порог превышается, то делается вывод что найденные компоненты не находятся в относительной близости и как следствие, что найденные области не являются текстом и соответственно субтитрами.

2.2 Определение мест смены текста

Для успешного создания временных меток, указывающих время начала и конца фрагмента текста субтитров, необходимо найти кадры, между которыми текст субтитров сменяется на иной или вообще исчезает полностью. Данная задача решается несколькими способами, например, сравнение текста, полученного при распознавании и сравнение количества общих точек двух последовательных кадров. Сравнение полученного текста является простым, но достаточно точным методом. У него есть несколько недостатков. Во-первых, при неправильном распознавании даже одного символа, полученный текст уже отличается. Заменив обычное посимвольное сравнение более сложным анализом расстояния между двумя строками подсчитанного методом анализа текста, который будет не чувствителен к замене одного символа, можно решить данную проблему, что тем не менее усложняет программу и требует время. Во-вторых, сравнение текста будет означать получение такового из каждого кадра, что сильно замедляет работу программы.

Его альтернатива, используемая в данном алгоритме, не имеет подобных проблем. После применения логического “И” к двум последовательным бинаризованым кадрам видео проводится подсчёт оставшихся точек, предположительно принадлежащих тексту. Сравним полученное число с количеством аналогичных пискелей, но уже только первого изображения. Данный процент показывает количество исчезнувших со сменой кадра точек, что на прямую показывает изменение текста, который данные пиксели отображают. В результате экспериментов было установлено, что смена текста происходит при потере более 40% точек, что является порогом для сравнения. Касательно скорости работы упоминаемой в альтернативном методе, применение логического “И” является значительно более быстрой операцией по сравнению с распознаванием. Время работы около 1 мс (миллисекунда), в то время, как распознавание требует около 400 мс (миллисекунд). Предобработка дополнительного кадра не учитывается в виду возможности использовать предыдущий обработанный кадр, что не тратит вычислительные ресурсы на повторную работу, как это и было реализовано

непосредственно в коде программы.

Определив места смены текста и имея классификатор его наличия на изображении, мы получаем возможность определения временных меток, которые являются обязательной частью субтитров. Данная информация позволяет делать вывод о нынешнем промежутке времени за счёт изучения предыдущего. Если кадр с текстом сменяется на "чистый" кадр, то данный момент во времени запоминается, что аналогично и в обратном случае появления субтитров. Последний третий случай запоминания времени, когда при существовании субтитров на двух последовательных кадрах, обнаруживается смена текста. В данном случае необходимо запоминание этого места дважды для разделения нового текста от старого. Таким образом, на каждый полученный текст создаётся по две временных метки, которые указывают на его границы, после чего достаточно взять их парами для соответствия полученным текстам. В начале же видео, для правильной работы алгоритма, считается что до этого момента субтитров не было.

2.3 Распознавание текста

Как и писалось выше в параграфе Метод Заливки, в результате предобработки необходимость в специальном методе исчезает, позволяя использовать предобученные методы рассчитанные на задаче распознавания чёрного текста на белом фоне. Для данной задачи был использован OCR engine Tesseract[13], в виду его открытой лицензии, кроссплатформенности, поддержки более чем 100 языков, удобства использования и в особенности его способность распознавать любой печатный шрифт[5], что идеально для распознавания субтитров.

Реализация и экспериментальные результаты

Код для решения поставленной задачи писался на Python 3 с использованием библиотеки OpenCV и для распознавания текста OCR Engine Tesseract. В итоге конечный алгоритм выглядит как на рис. 7. Также готовая реализация данного метода была выложена на GitHub[1], где любой желающий может скачать код для использования.

Конечным результатом работы алгоритма является, либо текст и временные метки возвращённые непосредственно во время работы программы, либо файл формата srt[11], который можно использовать в видео проигрывателях предварительно обработав по необходимости. Данный файл сформирован из данных полученных в результате работы алгоритма, а именно номера строки субтитров, временных границ и текста, расположенных в таковом порядке для каждой отдельной строки в соответствии с форматом srt.

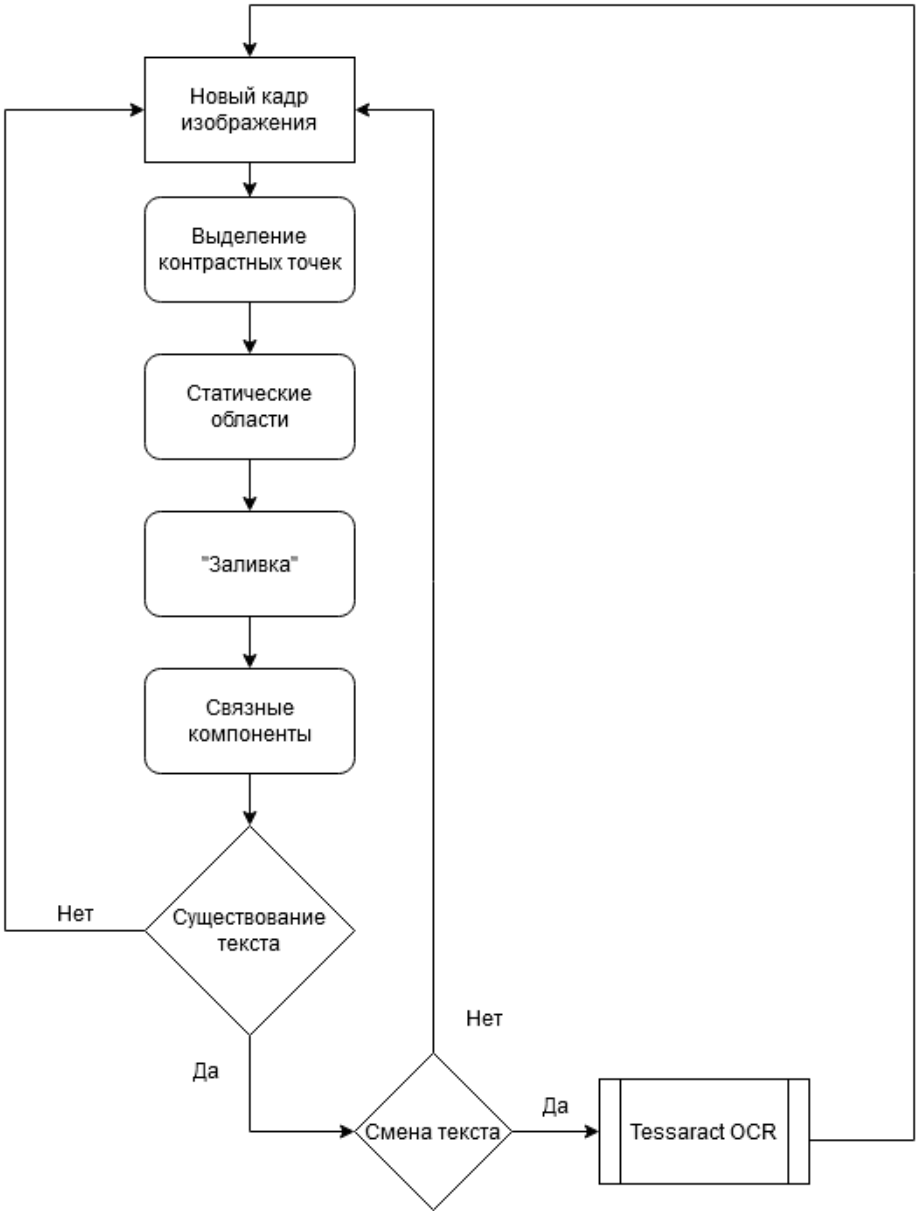


Рис. 7: Конечный алгоритм.

Для проверки распознавания были запущены конечный алгоритм дан-

Тип видеозаписи	Алгоритм/Программа	Размер текста	Доля пропущенных строк	Точность распознавания	Доля артефактов
Классический фильм	Искомый	Нормальный	0%	93,6%	0,6%
		Малый	4,7%	96,5%	0,1%
	Искомый - распознавание без предобработки	Нормальный	0%	63,3%	24,6%
		Малый	4,7%	43,6%	16,1%
	Videocr	Нормальный	22,7%	49,2%	0,5%
		Малый	30,2%	93,6%	4,2%
Высококонтрастная анимация	Искомый	Нормальный	2,5%	98,9%	0,4%
		Малый	15%	99,3%	1,1%
	Искомый - распознавание без предобработки	Нормальный	2,5%	34,3%	16,6%
		Малый	15%	40,7%	5,3%

Таблица 1: Сравнение результатов

ной работы и также, его аналог, но уже с применением алгоритма распознавания к искомому изображению, чтобы проверить улучшение качества распознавания, см. таблицу 1. Помимо этого, был запущен программный аналог videocr, выбранный среди своих конкурентов как наиболее автоматизированный и современный с последним обновлением 19 декабря 2019 года,помимо этого являющийся общедоступным пакетом для Python. Стоит отметить очень низкую скорость работы videocr, а именно анализ 10 секунд видео занимает 23 минуты, для сравнения описанный в данной работе алгоритм, при проверке на том же компьютере показал скорость работы в среднем 12 кадров видео в секунду, что для того же отрывка с частотой кадров 24 в секунду означает общее время работы в среднем 20-25 секунд, из-за периодических замедлений для распознавания. Все алгоритмы запускались со стандартным набором параметром, не меняющимся в зависимости от типа видео с целью получения показателя автоматизации методов. При рассмотрения videocr для распознавания на фоне высококонтрастной анимации, алгоритм не дал результатов, пропустив почти все субтитры и поэтому был убран из сравнения.

При подсчёте статистики ошибки были разделены на 2 типа: ошибки распознавания символов и ошибки обнаружения несуществующих символов. Если посмотреть на ошибки обнаружения несуществующих символов, то можно заметить схожесть с ошибкой второго рода, что, казалось бы, позволяет использовать более продвинутые метрики. Однако такие ошибки порождают новые символы, не существующие до этого в тестовом множестве, что не даёт нам использовать более сложные метрики как полноту

и F-меру в стандартном смысле. Поэтому был подсчитан процент создания новых ложных символов - артефактов от количества символов в оригинале для определения на сколько увеличился текст по сравнению с оригиналом. Как можно увидеть, также был проведено сравнение алгоритмов при разных размерах текста и посчитан процент пропущенных строк субтитров от искомого количества, показывающий работу алгоритма определения существования текста. Так как алгоритм данной работы от такового с анализом без предобработки отличаются только непосредственно в распознавании, тогда классификатор у них идентичный и соответственно результаты пропуска строк текста.

Помимо распознавания были проведены эксперименты с целью установления точности локализации субтитров. Было выделено 3 степени точности локализации: плохая, средняя и хорошая, где плохая не даёт конкретной позиции субтитров, средняя указывает область правильно с погрешностью около 15% и хорошая, точно указывающая на позицию текста в кадре, сравниваемые по площади пересечения идеальной локализации и полученной. Каждой степени было присвоено число, а именно плохо и 0, средняя и 0.5 , хорошо и 1, после чего подсчитано выборочное среднее, дабы установить среднее качество локализации, см. таблицу 2.

Тип видеозаписи	Размер шрифта	Оценка локализации
Классический фильм	Нормальный	0,893
Классический фильм	Малый	0,6
Контрастная анимация	Нормальный	0,778
Контрастная анимация	Малый	0,644

Таблица 2: Результаты локализации

Также во время проверки проверялся алгоритм определения смены текста и были получены такие результаты: даёт сбой в 4,25% случаев в живом видео и создал 6% от общего количества строк, что равносильно 1-2 новым повторным строкам субтитров. В случае же с анимацией процент сбоя равнялся 30%, а количество повторяющихся строк составило 70% процентов от общего количества, что приводило к мерцанию текста, а также излишним распознаваниям, повышающим шанс ошибки и замедляющим

работу алгоритма.

Выводы

Результаты экспериментов показали, что, данный алгоритм за счёт описанной выше предобрабоки, сумел значительно улучшить показатели алгоритма распознавания, применяемого к кадрам без предобработки в среднем на 50% точности, что видно в таблице 1. При анализе доли пропущенных строк, видно, что videocr несмотря на высокое качество распознавания игнорирует множество строк понижая его эффективность. Касательно лучших условий для алгоритма данной работы, взяв во внимание все данные, включая результаты проверки локализации и определения мест смены текста, можно сделать вывод что оптимальными условиями является субтитры нормального размер на живом фоне. Хоть и результаты распознавания показывают преимущество контрастной анимации, однако пропущенные строки портят данную статистику. Тем не менее алгоритм показал достойную работоспособность даже в таких суровых для него условиях.

Заключение

Получение субтитров, являющихся частью видеопотока – это комплексная задача, состоящая из множества этапов. С целью её решения были написаны программные решения в основном, полагающиеся на человеческий фактор. В данной работе было найдено автоматическое решение, которое при помощи средств обработки изображений использует особенности текста субтитров для выделения их среди фона любого вида. Оно включает в себя решение задачи шумоподавления, использованием контрастных точек в комбинации с статическими регионами кадров, окончательно очищенными и бинаризованными использованием полученного в результате проб и ошибок метода “заливки”. Реализация данных методов в неблагоприятной среде анимации также показала положительный результат в виде увеличения работоспособности программы за счёт контраста, причём независимо от него. Задачи локализации текста и символов субтитров с использованием вышеописанных средств решены методом поиска связных компонент изображения. Он позволяет не только найти символы текста вместе с ним, решая поставленные задачи, но и получить их рамку, дающую возможность использовать алгоритмы посимвольного распознавания текста, что даёт возможность модификации алгоритма, полученного в данной работе под необходимый метод распознавания. Применение логического “И” из статических областей для обнаружения смены текста кадров, оказалось чрезвычайно быстрым решением, которое избавило от необходимости распознавания каждого кадра. Это значительно ускорило работу программы и обеспечило возможность получить время показа субтитров.

Описанные выше идеи стали основой данной работы, позволив быстро и независимо от цвета текста, подготовить всё необходимое для последующего использования Tesseract, избежав недостатки других методов. Tesseract закончил алгоритм, качественно получая необходимый текст независимо от платформы, что вместе с реализацией на Python делает данную программу работоспособной на любой из основных операционных систем, таких как Windows, Linux, Mac OS и других UNIX-подобных системах. Впоследствии, при создании алгоритма устранения субтитров из видеопо-

тока, если мы дополним его предложенным автором решением, появится возможность замены встроенных переключаемыми субтитрами по желанию смотрящего. Подводя итог, данное исследование представило людям ещё один способ работы с видео, тем самым, расширяя наши возможности в восприятии видео и избавляя от необходимости ручной работы по переписыванию текста.

Список литературы

[1] Реализация алгоритма из данной работы. [Электронный ресурс]: https://github.com/WhiteSpirt25/SubtitlesExtractor

[2] BBC Subtitle Guidelines. [Электронный ресурс]: https://bbc.github.io/subtitle-guidelines/

[3] Burnt-in subtitle extractor. [Электронный ресурс]: https://github.com/roybaer/burnt-in-subtitle-extractor

[4] CCExtractor's home page. [Электронный ресурс]: https://www.ccextractor.org/start

[5] Comparison of optical character recognition software. [Электронный ресурс]: https://en.wikipedia.org/wiki/Comparison_of_optical_character_recognit 3

[6] Gaussian blur From Wikipedia, the free encyclopedia. [Электронный ресурс]: https://en.wikipedia.org/wiki/Gaussian_blur

[7] Jianfeng Xu, Shaofa Li. Caption location, tracking and enhancement in digital video // 2004 International Conference on Intelligent Mechatronics and Automation, 2004. Proceedings. : IEEE.

[8] Milyaev S. и др. Image Binarization for End-to-End Text Understanding in Natural Images // 2013 12th International Conference on Document Analysis and Recognition. : IEEE, 2013.

[9] OpenCV Documentation. [Электронный ресурс]: `https://docs.opencv.org/2.4/index.html`

[10] Optical Character Recognition (OCR) – How it works. [Электронный ресурс]: Nicomsoft.com.

[11] SRT File Format. [Электронный ресурс]: https://en.wikipedia.org/wiki/SubRip#File_format

[12] SubRip Official cite. [Электронный ресурс]: https://sourceforge.net/projects/subrip/

[13] Tesseract OCR. [Электронный ресурс]: https://github.com/tesseract-ocr/tesseract

[14] Videocr. [Электронный ресурс]: https://github.com/apm1467/videocr

[15] Xiaoou Tang и др. A spatial-temporal approach for video caption detection and recognition // IEEE Transactions on Neural Networks. 2002. Т. 13. № 4. С. 961–971.

[16] Yongjiu L. и др. Video Subtitle Location and Recognition Based on Edge Features // 2019 6th International Conference on Dependable Systems and Their Applications (DSA). : IEEE, 2020.

[17] Zafarifar B., Jingyue Cao, With P.H.N. de. Instantaneously responsive subtitle localization and classification for TV applications // IEEE Transactions on Consumer Electronics. 2011. Т. 57. № 1. С. 274–282.

[18] Zhu Y., Yao C., Bai X. Scene text detection and recognition: recent advances and future trends // Frontiers of Computer Science. 2015. Т. 10. № 1. С. 19–36.