МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ОТЧЕТ
ПО УЧЕБНОЙ ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ

Студент
гр. БИС-22-1 _____ В.В. Олейник

Руководитель
Канд. хим. наук, доцент
каф. ИТС _____ Б.К. Васильев

Владивосток 2024

МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на учебную ознакомительную практику

Студент гр. БИС-22-1 Олейник Виталий Владимирович

1 Тема работы: Разработка приложения для работы с видеопотоком OpenCV

2 Срок сдачи работы: 13.07.2024.

3 Содержание задания

3.1 Решение практических задач

Задача 1.

Изучить библиотеку OpenCV для работы с видеопотоками.

Задача 2.

С использованием языка программирования Python, создать кроссплатформенное приложение с графическим интерфейсом для управления фильтрами видеопотока. Обязательно обеспечить версию для Linux.

3.2 Оформление и защита отчета

Отчет по учебной исследовательской практике должен содержать: титульный лист; индивидуальное задание; содержание; цель и задачи; описание выполненных заданий (в соответствии с методическим руководством и требованиями преподавателя); выводы и предложения; список использованных источников; графический материал (схемы, графики, технологические карты).

Календарный план-график работ прилагается к отчету отдельным листом.

Требования к оформлению отчетов.

Отчёт предоставляется в печатном виде с выполнением требований нормоконтроля и состоит из следующих разделов:

Введение. Во введении обосновывается цель и задачи прохождения практики.

В разделе 1 выполняется краткий обзор литературы по теме индивидуального задания, подбирается необходимый математический инструментарий (аппарат), производится его анализ и разрабатывается алгоритм решения задачи.

В разделе 2 осуществляется обоснование выбора среды реализации разработанного алгоритма.

В разделе 3 описывается процесс кодирования в выбранной среде и языке программирования и основные элементы управления создаваемым приложением.

В разделе 4 описывается процесс тестированию и отладки программного кода.

Раздел 5 (рекомендуемый) должен содержать краткое иллюстрированное руководство пользователя.

Заключение. В заключении обобщается изложенный в отчете материал, делаются выводы.

Объем отчёта составляет 20 страниц.

Отчёт по практике оформляется в соответствии с «Требованиями к оформлению текстовой части выпускных квалификационных работ, курсовых работ (проектов), рефератов, контрольных работ, отчётов по практикам, лабораторным работам (СК-СТО-ТР04-1.005-2020)». Для оформления графического материала блок-схем, алгоритмов использовать ГОСТ 19.701-90 который распространяется на условные обозначения (символы) в схемах алгоритмов, программ, данных и систем и устанавливает правила выполнения схем, применяемых для отображения различных видов задач обработки данных и средств их решения.

Руководитель
Канд. хим. наук, доцент
каф. ИТС _____Б.К. Васильев

Задание получил: _____В.В. Олейник

МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

КАЛЕНДАРНЫЙ ПЛАН-ГРАФИК (ДНЕВНИК)

прохождения учебной ознакомительной практики студента ВВГУ

Студент Олейник Виталий Владимирович направляется для прохождения учебной ознакомительной практики «Учебная ознакомительной практика» в ВВГУ, кафедра ИТС, г. Владивосток.

С __10.06.2024__ по __13.07.2024__ г.

№ п/п	Содержание выполняемых работ по программе	Сроки выполнения		Заключение и оценка руководителя	Подпись руководителя
		Начало	Окончание		
1	**Задание 1.** Анализ поставленной задачи	10.06.2024	17.06.2024		
2	**Задание 2.** Сбор и анализ информации	18.06.2024	19.06.2024		
3	**Задание 3.** Разработка решения поставленных задач	20.06.2024	10.07.2024		
4	**Задание 4.** Оформить отчет и документы практики в печатном и электронном виде и представить на защиту	11.07.2024	13.07.2024		

Согласовано:

Студент-практикант_____ Олейник В.В.
подпись

Руководитель от кафедры _____ Васильев Б.К.
подпись

дата

Содержание

Введение 3
1. Приложение чтения видеопотока 4
1.1 История 4
1.2 Понятие видеопотока 5
1.3 Алгоритм создания приложения 6
2. Выбор языка программирования 8
3. Процесс написания программного кода 11
4. Тестирование программного кода 12
Заключение 14
Список использованных источников 17

Введение

В рамках данной учебной практики необходимо разработать программу использующую библиотеку OpenCV для работы с видеопотоками и инструментов, доступных в операционной системе Linux. Основной целью задания является создание интерактивного графического интерфейса пользователя (GUI), который позволяет подключаться к ip камерам, выводить несколько видеопотоков.

OpenCV является бесплатной библиотекой с открытым исходным кодом, которая предоставляет широкий спектр инструментов для обработки изображений и видео. Благодаря своей доступности и поддержке множеств языков программирования таких как Python, C++ и Java, OpenCV стал используется как в научных исследованиях, так и в коммерческих нуждах.

Первая задача заключается в изучении библиотеки OpenCV (Open Source Computer Vision Library), которая является мощным инструментом для разработки приложений в области компьютерного зрения и обработки изображений и видео. Глубокое понимание возможностей OpenCV является фундаментальным этапом, необходимым для успешной реализации проектов по работе с видеопотоками.

Вторая задача состоит в создании программы, способной обрабатывать и отображать их содержимое в реальном времени. Цель этой задачи - разработка интерактивного приложения, которое позволяет захватывать, обрабатывать и отображать видеопотоки с возможностью настройки параметров отображения.

Проект включает в себя глубокое изучение области компьютерного зрения компиляции Выбор языка программирования python обуславливается широким набору библиотек и инструментов, обширной документации и поддержки сообществом.

1. Приложение чтения видеопотока

1.1 История

OpenCV была создана в 1999 году исследовательской группой компании Intel. Целью разработки было предоставить стандартные инструменты для создания приложений компьютерного зрения, которые могли бы использоваться в различных областях, таких как робототехника, безопасность, медицина и развлечения.

Первая версия библиотеки была выпущена для внутреннего использования в intel, а затем стала доступной для широкой общественности в качестве проекта с открытым исходным кодом

В 2000 году была выпущена первая публичная версия OpenCV. Библиотека быстро приобрела популярность среди своей производительности, простоте использования и обширной документации. В этот период библиотека поддерживалась и развивалась в основном инженерами Intel, которые добавляли новые функции и улучшения.

В 2006 году OpenCV получила значительную поддержку от компании Willow Garage, известной своими разработками области робототехники. Willow Garage активно инвестировала в развитие OpenCV и способствовала расширению ее функциональности. В этот период были введены многие ключевые функции и модули, такие как функции для машинного обучения, детекции объектов распознавания лиц и работы с 3D- изображениями.

В 2012 году была основана организация OpenCV.org, которая взяла на себя ответственность за дальнейшее развитие и поддержку библиотеки. Эта организация обеспечила стабильное развитие проекта и продолжила работы над улучшением и расширением функциональности OpenCV. В этот период началась активная работа над интеграцией OpenCv с другими популярными библиотеками и фреймворками, такими как TensorFlow, PyTorch и другие.

В июне 2015 года была выпущена версия OpenCV 3.0. Эта версия включала множество новых функций и улучшений, таких как поддержка параллельных вычислений, улучшение производительности и новые алгоритмы машинного обучения. Введения модуля OpenCV Contrib позволило сообществу вносить вклад в развитие библиотеки, добавляя новые функции и модули, что значительно расширило возможности OpenCV.

В ноябре 2018 года была выпущена версия OpenCV 4.0. В этой версии были проведены значительные изменения в архитектуре библиотеки, что значительно улучшило ее производительность и удобство использования. Были добавлены новые алгоритмы и улучшены существующие, а также внедрена поддержка современных технологий, таких как глубокое обучение и нейронные сети.

В последние годы OpenCV продолжает активно развиваться, внедряя новейшие достижения в области компьютерного зрения и машинного обучения. Библиотека поддерживает работу с нейронными сетями, распознавание объектов в реальном времени, обработку больших данных и многие другие передовые технологии. OpenCV используется в широком спектре приложений, включая автономные автомобили, робототехники, системы безопасности медицинскую диагностику и развлечения.

1.2 Понятие видеопотока

Синтез речи – это технология, которая помогает озвучивать текст. Иногда вместо термина «синтез речи» используют аббревиатуру TTS, которая расшифровывается как Text-to-Speech. Есть множество методов синтеза речи, но наибольшее значение имеют две группы технологий: конкатенация и синтез по параметрам с использованием глубоких нейронных сетей. При конкатенации используются предварительно записанные звуки, слова и фразы. Это огромный массив записей, ведь звучание каждого фрагмента зависит от фонетического окружения. Синтезированная речь получается довольно качественной, хотя и с некоторыми недостатками: монотонностью и артефактами на стыке фрагментов. Синтез на основе глубокого обучения стал активно развиваться примерно с 2016 года. Сегодня созданная этим методом речь практически не отличается от натуральной.

Чтобы подготовить TTS-систему, нужен датасет для обучения нейросети. Сначала создаются аудиозаписи с разными голосами и темами, дальше они обрабатываются разметчиками и превращаются в датасет, на котором сеть и обучается. Печатный текст состоит из букв, слов, а аудиозапись можно разобрать на мелкие фрагменты, каждый из которых имеет свой «рисунок» на дорожке аудиозаписи – спектрограмме. Задача нейросети в том, чтобы сначала научиться устанавливать соответствие «рисунка» аудиозаписи тексту, а потом научиться воспроизводить текст по этим примерам самостоятельно.

Чтобы синтезированная речь не была роботоподобной, к обучению подключаются дополнительные модели: например, модель, которая обучается предсказанию, где по тексту нужно сделать смысловую паузу, или модель, которая работает над интонацией (создаёт в нужных местах повышение-понижение тона и громкости).

Для более человекоподобной речи также важно, чтобы алгоритм синтеза включал в себя понимание контекста. В этом случае конечная интонация исходит не только из знаков препинания и пауз, а ещё и из смысла текста [2].

Таким образом, синтез речи (Text-to-Speech, TTS) – это технология, позволяющая озвучивать текст. Существуют две основных группы методов синтеза речи: конкатенация и синтез на основе глубоких нейронных сетей. При конкатенации используются

предварительно записанные звуки, что приводит к качественной, но часто монотонной речи. С 2016 года активно развивается синтез на основе глубокого обучения, создающий почти натуральную речь. Для создания TTS-системы необходим датасет аудиозаписей, соответствующий тексту. Нейросеть учится сопоставлять аудиофрагменты и текст, а также воспроизводить их. Дополнительные модели предсказывают паузы и интонации, что делает синтезированную речь более человекоподобной и учитывающей контекст текста.

1.3 Алгоритм создания приложения

Задача 1: Определение целей и требований приложения

Разработать приложение использующее библиотеку OpenCV(open source computer Vision Library). Разработка структуры программы, учитывая необходимость использовать многопоточность для одновременного чтения с нескольких источников

Задача 2: настройка проекта и зависимостей

Создание структуры проекта, определив основные модули, установка необходимых зависимостей, таких как OpenCV, NumPy, ZeroMQ и другие.

Задача 3: настройка конфигурации

Создание файла конфигурации, где определены адреса и другие параметры конфигурации, которые могут изменяться в зависимости от среды выполнения

Задача 4: разработка основной логики программы

Использование “argparse” для обработки аргументов командной строки, таких как количество камер, включение/отключение изображения, фильтрация на основе пользователя.

Задача 5: реализация класса “VideoStream”

Создание класса “VideoStream”, который будет отвечать за управление потоками видео.Инициализация класса с параметрами: адреса камер, включение изображения, размер изображения и необходимости использовать ZeroMQ. В методе “run()” использовать многопоточность для запуска функции “_create_stream()”.

Задача 6: обработка кадров и извлечение ключевых кадров

Создание класса “KeyframeExtractor” для фильтрации кадров на основе различных методов: абсолютной разницы и структурного сходства изображений. Реализация Метода для расчета пороговых значений и сравнения кадров с пороговыми значениями для выявления ключевых кадров

Задача 7: тестирование программы

Проведение тестирования на различных сценариях, включая чтение видеопотоков с разных камер, фильтрация кадров и отображение на экране. Проверка корректности

фильтрации кадров и извлечение ключевых кадров в зависимости от установленных параметров.

Задача 8: оптимизация и профилирование

Проведение профилирования для выявления узких мест и оптимизации производительности , особенно в контексте работы с камерами и многопоточности. Оптимизация кода для улучшения скорости работы программы и эффективности использования ресурсов.

Задача 9: Завершение и внедрение

Завершение разработки, проверка выполнения всех требований.Внедрение программы в целевуй среду.

Этот алгоритм охватывает основные этапы разработки программы для работы с видеопотоком, начиная от настройки окружения и определения требований до завершения разработки и внедрения в реальную среду.

2. Выбор языка программирования

Выбор языка программирования практически сразу пал в пользу Python, ведь он является компилируемым языком, и его программы обычно выполняются быстрее, чем программы, написанные на интерпретируемых языках, таких как Python или JavaScript.

В критичных к производительности задачах, таких как обработка аудио и синтез речи, скорость выполнения кода играет важную роль.

C++ предоставляет тонкий контроль над памятью и позволяет разработчикам управлять системными ресурсами напрямую, что важно для управления аудиофайлами, буферами данных и другими аспектами низкоуровневого программирования.

Возможность использования указателей и ручного управления памятью позволяет оптимизировать использование ресурсов и улучшить производительность.

C++ поддерживает множество библиотек и фреймворков, необходимых для нашего проекта, таких как FLTK для создания графического интерфейса пользователя (GUI), “espeak-ng”, “festival”, “RHVoice” для синтеза речи и `ffmpeg` для работы с аудиофайлами.

Наличие статических и динамических библиотек позволяет интегрировать и использовать существующие решения для синтеза и обработки аудио. Язык программирования C++ легко взаимодействует с операционной системой Linux и поддерживает использование системных вызовов и API, что облегчает реализацию функций работы с файлами, управления процессами и межпроцессного взаимодействия.

Прямая работа с системными интерфейсами позволяет более эффективно использовать возможности операционной системы.C++ обладает богатым наследием и активным сообществом разработчиков, что облегчает поиск решений для возникающих проблем и доступ к большому количеству документации, учебных материалов и примеров кода [5].

Рассмотрим преимущества python перед другими языками:

1. Python и C++ .

Простота и читабельность: C++ имеет значительно более сложный синтаксис, нежели Python, это делает c++ более громоздким и менее читабельным, что усложняет поддержку и расширение проектов.

Машинное и глубокое обучение: Python является основным языком для библиотек машинного и глубокого обучения. Python позволяет легко интегрировать OpenCV с моделями машинного и глубокого обучения, что делает его предпочтительным выбором для таких задач.

2. Python и Java

экосистема библиотек: Python имеет богатую экосистему библиотек, которые легко интегрируются с OpenCV, в то время как у java библиотек для компьютерного зрения и машинного обучения меньше и они хуже развиты.

Активное сообщество и поддержка: хоть java и имеет большое и активное сообщество, однако ресурсы и примеры кода для интеграции с OpenCV меньше, чем на Python.

3. Python и C.

Использование в индустрии:Python используется в различных отраслях, включая науку о данных, машинное обучение, веб-разработку и автоматизацию. Компании и стартапы предпочитают Python за его гибкость и скорость разработки. Язык программирования C же в свою очередь активно используется в системном программировании и разработке операционных систем .

Управление памятью: в то время как в Python управление памятью происходит автоматически, в языке программирования C управление памятью происходит вручную, что может замедлить процесс работы, но при этом, C дает разработчикам больше контроля, но также требует от них больше внимания и аккуратности.

Ниже, в таблице, представлено наглядное сравнение некоторых языков программирования.

Таблица 1 – Сравнительный анализ языков программирования

Критерий	Python	C++	Java	C
Простота и читабельность	простота и читабельность кода, легкость изучения	сложный синтаксис, требуются детальные знания	умеренная сложность	сложный синтаксис, требуется глубокое понимание
экосистема библиотек	широкий набор библиотек	много библиотек, особенно для высокопроизводительных задач	ограниченный набор библиотек для компьютерного зрения	ограниченное количество библиотек

Критерий	Python	C++	Java	C

производительность	средняя производительность	высокая производительность оптимизация под низкоуровневый язык	средняя производительность	высокая производительность, сложность оптимизации
машинное и глубокое обучение	легкая интеграция	поддержка через внешние библиотеки, сложнее интеграция	ограниченная поддержка, сложная интеграция	сложная интеграция, требует много усилий
активное сообщество и поддержка	очень активное сообщество, обширная документация	Активное сообщество, много ресурсов	Активное сообщество, меньше ресурсов по OpenCV	Менее активное сообщество, ограниченные ресурсы
Управление памятью	Автоматическое управление памятью	ручное управление памятью, высокая гибкость	Автоматическое управление памятью	Ручное управление памятью, высокая гибкость
Использование в индустрии	Широко используется в науке, образовании, стартапах	Широко используется в высокопроизводительных и критически важных приложениях	Используется корпоративных и мобильных приложениях	Используется в системах низкого уровня, встраиваемых в системах

Как видно из таблицы, Python является оптимальным выбором для проекта посвященному работе с библиотекой OpenCV, благодаря своему богатству библиотек, количества материала по теме, а также простоте в освоении Это делает Python лучшим выбором для реализации данного проекта.

3. Процесс написания программного кода

При разработке данного кода было несколько ключевых этапов, которые были выполнены для создания функционального приложения для работы с видеопотоками

Изначально была проведена структурная проработка приложения. Я определил необходимые модули и классы, которые потребуются для управления видеопотоками и извлечения ключевых кадров. Это позволило сделать код более организованным и легко расширяемым в будущем.

Для удобства управления конфигурационными данными, такими как адреса камер или файлов видео, был использован “.env” файл. это обеспечило централизованное хранение параметров конфигурации их изменение их внутри кода.

Основной скрипт “main.py” был настроен как точка входа в приложение. С помощью библиотеки “argparse” была добавлена поддержка аргументов командной строки, что позволяет пользователю легко настраивать количество источников видеопотоков, включение отображения потоков и другие параметры.

Модуль “creating_video_stream.py” содержит класс “VideoStream”, который управляет захватом видео, обработкой кадров и, при необходимости, их отображением. Использование “multiprocessing.Pool” позволило создать несколько параллельных процессов для обработки каждого источника видео, обеспечивая тем самым эффективную работу с множеством камер или файлов.

Также в этом модуле реализованы методы для создания объекта ZroMQ (“create_zmq()”), который обеспечивает межпроцессное взаимодействие, и для извлечения кадров по временному интервалу (“_extract_frames()”, что может быть полезно для дальнейшего анализа видеоданных.

в конечном итоге разработка кода включила в себя не только написание конкретных функций и методов, но и предварительное проектирование структуры приложения, настройку окружения для хранения конфигурационных данных и обеспечение легкости расширения функциональности в будущем

4. Тестирование программного кода

Тестирование кода, который управляет видеопотоками и извлекает ключевые кадры, оказалось значительным вызовом на пути к разработке стабильной системы. Этот процесс требовал не только глубокого технического понимания, но и организационных усилий для обеспечения полного покрытия всех аспектов работы с медиа данными и многопоточностью

Тестирование было затруднено несколькими факторами. Во-первых, код оперирует внешними источниками, такими как камеры или файлы видео. Это создавало зависимость от доступности ресурсов во время тестирования, что могло привести к непредсказуемым результатам при недоступности камер или изменениях в содержимом видеофайлов.

Управление медиа-данными также представляло собой вызов, требующий точного контроля над обработкой видеофреймов и извлечением ключевых кадров. Ошибки в обработке или их масштабного могли негативно сказаться на результате тестирования.

Еще одной сложностью сложностью стало использование многопоточности с помощью “multiprocessing .Pool”. Это требовало Тщательной синхронизации и управления доступом к ресурсам между процессами через ZeroMQ добавляла дополнительные сложности, так как требовала специфического окружения для проверки визуальной корректности отображаемых данных.

Несмотря на всю сложность тестирования, одной из основных проблем была нехватка времени. Не всегда удавалось полностью охватить все сценарии и задачи тестирования из-за ограниченных ресурсов и времени. Это могло привести к упущению потенциальных проблем или недоработок, которые могли проявиться только в специфических условиях.

Для успешного тестирования кода были применены различные методы. Модульное тестирование позволяло проверять отдельные компоненты кода на изолированных данных, что помогало выявлять базовые ошибки в работе с медиа данными.

Интеграционное тестирование использовалось для проверки взаимодействия между компонентами системы, особенно между процессами, работающими с различными видеопотоками. Это включало проверку правильности передачи данных и корректности их обработки.

Функциональное тестирование проверяет выполнение функциональных требований системы в целом, включая извлечение ключевых кадров при различных условиях видео. Это важно для обеспечения работы системы с высокой надежностью и точностью в различных сценариях использования.

Тестирование производительности позволяло оценить скорость выполнения основных операций, таких как обработка кадров и извлечение данных. Это было важно для

оптимизации работы кода и предотвращения проблем с производительностью в реальном времени.

Тестирование кода для работы с видеопотоками было сложным и многогранным процессом, который требовал систематического подхода и внимания к каждой детали. Несмотря на вызов, успешное тестирование позволило выявить и исправить множество потенциальных проблем, обеспечивая стабильную и эффективную работу программы

Одним из ключевых аспектов разработки была реализация функционала конвертации текста в аудиоформаты с использованием внешней библиотеки ffmpeg. Этот процесс включал создание кода для взаимодействия с ffmpeg, а также управление процессами конвертации и обработки аудиоданных. Особое внимание уделялось точной настройке параметров конвертации, необходимой для обеспечения высокого качества и эффективности процесса. Это включало выбор оптимальных настроек для кодеков, битрейтов и других параметров, чтобы гарантировать соответствие требованиям качества и размера файлов аудиоформатов.

Для работы с текстовыми данными программа предусматривала функционал загрузки текста из файлов различных форматов. Важной задачей стало обеспечение разделения текста на блоки, учитывая ограничения на длину текста, которые могли быть наложены используемым сервисом синтеза речи. Разработка алгоритма автоматического разбиения и объединения текстовых блоков в один аудиофайл требовала тщательного проектирования и тестирования.

Заключение

Разработка программы для работы с видеопотоками с использованием OpenCV и других вспомогательных библиотек стала сложным и многоэтапным процессом. Этот проект предоставил возможность погрузиться в глубины компьютерного зрения, многопоточности и межпроцессного взаимодействия, требуя от разработчиков значительных усилий на каждом этапе.

Изначально была проведена структурная проработка приложения. Это включало определение необходимых модулей и классов, которые будут управлять видеопотоками и извлекать ключевые кадры. Благодаря этому подходу, удалось создать организованный и легко расширяемый код, что является важным аспектом для будущего развития и поддержки проекта. Для управления конфигурационными данными, такими как адреса камер или видеофайлов, был использован файл ".env". Это решение обеспечило централизованное хранение параметров конфигурации, что упростило их изменение без необходимости внесения правок непосредственно в код.

Основной скрипт "main.py" был настроен как точка входа в приложение. Использование библиотеки "argparse" позволило добавить поддержку аргументов командной строки, что значительно повысило гибкость приложения. Пользователи могут легко настраивать количество источников видеопотоков, включение отображения потоков и другие параметры, что делает приложение более универсальным и удобным в использовании. Модуль "creating_video_stream.py" содержит класс "VideoStream", который отвечает за захват видео, обработку кадров и, при необходимости, их отображение. Применение "multiprocessing.Pool" позволило создать несколько параллельных процессов для обработки каждого источника видео, что обеспечило эффективную работу с множеством камер или видеофайлов. Это решение значительно повысило производительность и масштабируемость приложения.

В процессе разработки также были реализованы методы для создания объекта ZeroMQ ("create_zmq()"), который обеспечивает межпроцессное взаимодействие, и для извлечения кадров по временному интервалу ("_extract_frames()"). Эти функции оказались особенно полезными для дальнейшего анализа видеоданных, обеспечивая гибкость и расширяемость приложения. Разработка кода включала в себя не только написание конкретных функций и методов, но и предварительное проектирование структуры приложения, настройку окружения для хранения конфигурационных данных и обеспечение легкости расширения функциональности в будущем.

Тестирование программного кода стало важной и сложной задачей. Код, управляющий видеопотоками и извлекающий ключевые кадры, требовал глубокого технического понимания и организационных усилий для обеспечения полного покрытия всех аспектов работы с медиа данными и многопоточностью. Во-первых, код оперирует внешними источниками, такими как камеры или видеофайлы. Это создавало зависимость от доступности ресурсов во время тестирования, что могло привести к непредсказуемым результатам при недоступности камер или изменениях в содержимом видеофайлов. Управление медиа-данными также представляло собой вызов, требующий точного контроля над обработкой видеофреймов и извлечением ключевых кадров. Ошибки в обработке или масштабировании могли негативно сказаться на результате тестирования.

Использование многопоточности с помощью "multiprocessing.Pool" добавляло дополнительные сложности, так как требовало тщательной синхронизации и управления доступом к ресурсам между процессами. Внедрение ZeroMQ для межпроцессного взаимодействия также требовало специфического окружения для проверки визуальной корректности отображаемых данных. Несмотря на всю сложность тестирования, одной из основных проблем стала нехватка времени. Не всегда удавалось полностью охватить все сценарии и задачи тестирования из-за ограниченных ресурсов и времени. Это могло привести к упущению потенциальных проблем или недоработок, которые могли проявиться только в специфических условиях.

Для успешного тестирования кода были применены различные методы. Модульное тестирование позволяло проверять отдельные компоненты кода на изолированных данных, что помогало выявлять базовые ошибки в работе с медиа данными. Интеграционное тестирование использовалось для проверки взаимодействия между компонентами системы, особенно между процессами, работающими с различными видеопотоками. Это включало проверку правильности передачи данных и корректности их обработки. Функциональное тестирование проверяло выполнение функциональных требований системы в целом, включая извлечение ключевых кадров при различных условиях видео. Это важно для обеспечения работы системы с высокой надежностью и точностью в различных сценариях использования. Тестирование производительности позволяло оценить скорость выполнения основных операций, таких как обработка кадров и извлечение данных. Это было важно для оптимизации работы кода и предотвращения проблем с производительностью в реальном времени.

Одним из ключевых аспектов разработки была реализация функционала конвертации текста в аудиоформаты с использованием внешней библиотеки ffmpeg. Этот процесс включал создание кода для взаимодействия с ffmpeg, а также управление процессами конвертации и обработки аудиоданных. Особое внимание уделялось точной настройке параметров конвертации, необходимой для обеспечения высокого качества и эффективности процесса. Это включало выбор оптимальных настроек для кодеков, битрейтов и других параметров, чтобы гарантировать соответствие требованиям качества и размера файлов аудиоформатов.

Для работы с текстовыми данными программа предусматривала функционал загрузки текста из файлов различных форматов. Важной задачей стало обеспечение разделения текста на блоки, учитывая ограничения на длину текста, которые могли быть наложены используемым сервисом синтеза речи. Разработка алгоритма автоматического разбиения и объединения текстовых блоков в один аудиофайл требовала тщательного проектирования и тестирования.

Тестирование кода для работы с видеопотоками было сложным и многогранным процессом, который требовал систематического подхода и внимания к каждой детали. Несмотря на вызов, успешное тестирование позволило выявить и исправить множество потенциальных проблем, обеспечивая стабильную и эффективную работу программы. В целом, проект продемонстрировал важность тщательного планирования, тестирования и отладки в процессе разработки сложных программных систем. Успешная реализация всех поставленных задач позволила создать функциональное и надежное приложение, способное эффективно работать с множеством видеопотоков и обеспечивать высокое качество обработки данных.

Список использованных источников

1 OpenCV: [сайт]. – URL: https://blog.skillfactory.ru/glossary/opencv (дата обращения: 10.07.2024)

2 OpenCV: [сайт]. – URL: https://opencv.org/ (дата обращения: 10.07.2024)

3 OpenCV программы. [сайт]. – URL: https://habr.com/ru/articles/519454/ (дата обращения: 10.07.2024)

4 OpenCV: [сайт]. – URL: https://skillbox.ru/media/code/kompyuternoe-zrenie-opencv-gde-primenyaetsya-i-kak-rabotaet-v-python/ (дата обращения: 10.07.2024)

5 OpenCV: [сайт]. – URL: https://vc.ru/u/1389654-machine-learning/661520-osvoenie-opencv-s-pomoshyu-python-polnoe-rukovodstvo-po-obrabotke-izobrazhenii-i-kompyuternomu-zreniyu (дата обращения: 10.07.2024)

6 OpenCV: [сайт]. – URL: https://pypi.org/project/opencv-python/ (дата обращения: 10.07.2024)

7 OpenCV Python: [сайт]. – URL: https://waksoft.susu.ru/2021/05/26/chtenie-i-zapis-video-s-ispolzovaniem-opencv/ (дата обращения: 10.07.2024)

8 OpenCV чтение видеопотока: [сайт]. – URL: https://ru.stackoverflow.com/questions/1272441/opencv-python(дата обращения: 10.07.2024)

9 Базовые операции с видео в OpenCV: [сайт]. – URL: https://robx.org/wiki/prog/opencv-biblioteka/opencv-video-editor/ (дата обращения: 10.07.2024)

10 OpenCV wiki Английский: [сайт]. – URL: https://en.wikipedia.org/wiki/OpenCV (дата обращения: 10.07.2024)

11 OPenCV wiki русский язык: [сайт] - URL: https://ru.wikipedia.org/wiki/OpenCV

12 понятие видеопотока : [сайт] - URL https://ru.wikipedia.org/wiki/цифровое_видео

13 FFmpeg: [сайт] - URL https://ru.wikipedia.org/wiki/FFmpeg

14 ZeroMQ: [сайт] - URl https://zeromq.org/languages/python/