МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Направление подготовки/специальность
09.03.01 Информатика и вычислительная техника

направленность (профиль)/специализация
«Технологии разработки программного обеспечения»

Выпускная квалификационная работа

Распознавание жестов при управлении презентацией на основе компьютерного зрения

Обучающегося 4 курса
очной формы обучения
Васильевой Марины Андреевны

Руководитель выпускной квалификационной работы:
кандидат физико-математических наук,
доцент кафедры ИТиЭО
Власов Дмитрий Викторович

Санкт-Петербург
2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время, в мире цифровых технологий и интерактивных решений потребность в удобных и интуитивно понятных способах взаимодействия с компьютерными системами возрастает. От современных компьютерных и робототехнических систем, управляемых человеком, требуют, как минимум, удобства и интуитивности использования, но основным критерием является скорость, с которой технология позволяет работать оператору.

В сфере презентаций необходимо обеспечить плавное управление слайдами, но в тоже время важно использовать простые и понятные инструменты, позволяющие сосредоточиться на содержании презентации, а не на ее технических аспектах управления. Это подчеркивается в статье «30 правил и секретов для успешной презентации»[6].

Жестовый и голосовой интерфейс делает выступления с презентациями более динамичными и выводит их на новый уровень – теперь они станут еще эффектнее и оригинальнее. Кроме того, технологии компьютерного зрения и машинного обучения позволяют реализовать точное распознавание жестов даже на стандартных устройствах с веб-камерами.

Объект исследования – процесс управления презентацией с использованием жестов.

Предмет исследования – разработка программы на Python с применением библиотек компьютерного зрения (OpenCV, MediaPipe) для распознавания жестов и управления слайдами.

Цель исследования – создание программного обеспечения, позволяющего управлять презентацией с помощью жестов, что повысит удобство и интерактивность выступлений.

В рамках работы выделены следующие задачи исследования:

- изучение научной литературы, статьей и существующих решений в области распознавания жестов;
- осуществление отбора инструментов для реализации проекта;
- проектирование архитектуры программы;
- реализация прототипа с графическим интерфейсом.

Практическая значимость работы заключается в создании доступного инструмента, который позволяет:

- упростить процесс управления презентацией, делая его более естественным;
- сократить зависимость от дополнительных устройств (кликеров, мыши);
- повысить вовлеченность аудитории за счет динамичного взаимодействия.

При выполнении работы использовались учебные и справочная литература, научные статьи, документация библиотек, а также открытые электронные ресурсы Интернет.

ГЛАВА 1. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ

1.1 Компьютерное зрение

Согласно [19], компьютерное зрение (Computer Vision, сокращенно CV) – это технология, которая помогает компьютерам анализировать визуальную информацию: распознавать объекты, лица, жесты, текст и даже эмоции.

За последние годы компьютерное зрение стало ключевой технологией во многих областях. Поэтому выделяют следующие задачи, которая оно решает [5]:

- Классификация изображений – определение категории или метки для всего изображения. Например, распознавание, является ли изображение автобусом или машиной.
- Детектирование объектов – нахождение и обозначение границ объектов на изображении, например, выделение автомобилей, людей или животных на улице.
- Сегментация изображений – разбиение изображения на семантические области (пиксельная классификация). Например, выделение каждого пикселя изображения, принадлежащего человеку, дороге или небу.
- Распознавание лиц – идентификация или верификация лиц на изображениях, например, разблокировка смартфона по лицу пользователя.
- Оценка позы человека – определение положения частей тела человека на изображении, например, определение координат рук, ног и головы.
- Генерация изображений – создание новых изображений или модификация существующих. Например, преобразование карандашного рисунка в цветное изображение.

- Видеоанализ – анализ последовательности кадров для выявления событий или трендов. Например, распознавание действий человека: бег, прыжок.
- 3D-восстановление – создание трехмерной модели объекта или пространства на основе изображений, например, создание 3D-модели здания по фотографиям.
- Отслеживание объектов – отслеживание движения объекта во времени.
- Распознание текста на изображениях, например, сканирование документов или номерных знаков.
- Сравнение изображений – определение схожести между двумя или более изображениями, например, проверка подлинности документов.
- Постановка ключевых точек – определение важных точек на объекте, например, глаза, нос, колени для создания анимированных аватаров.

Последовательность операций для получения конечного результата решения задачи компьютерного зрения может иметь вид [5], показанный на рисунке 1.

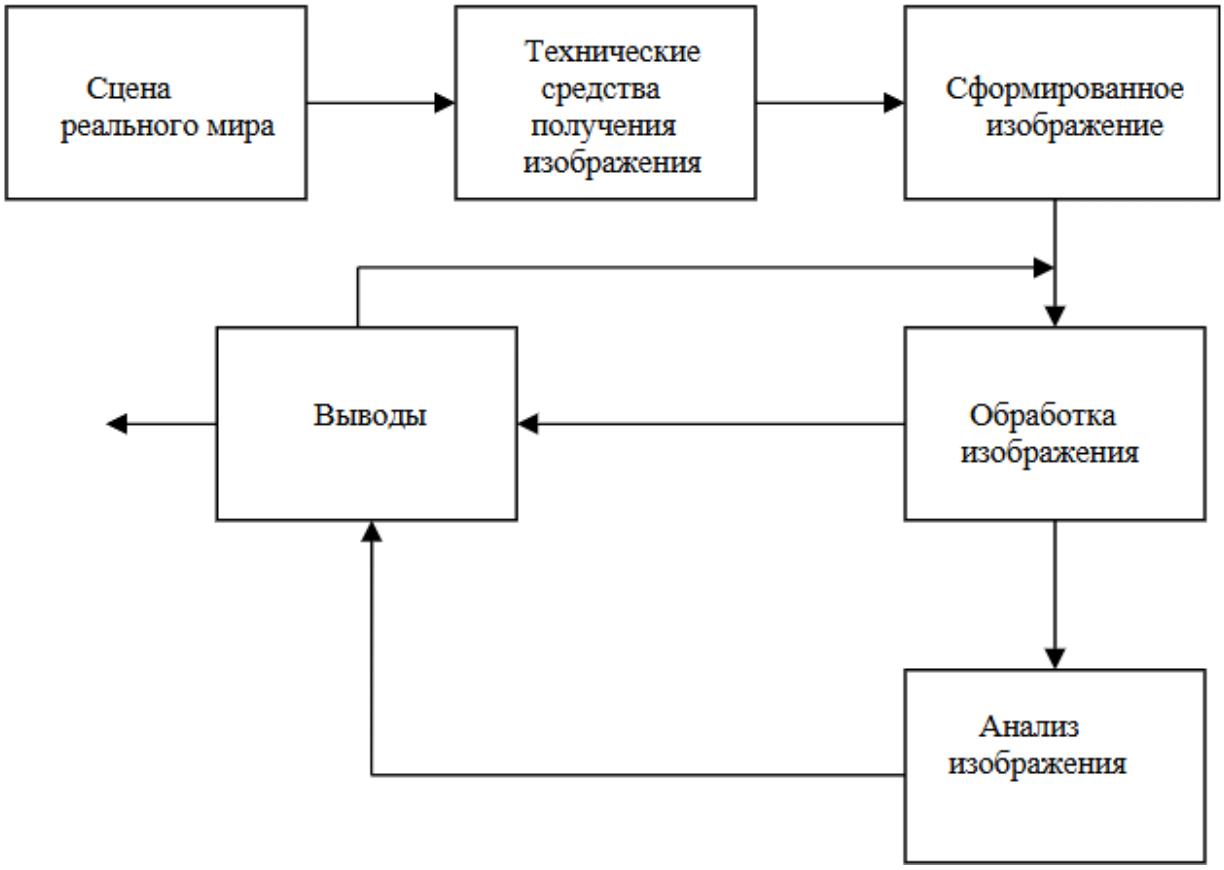


Рисунок 1 – Схема конвейера решения задачи компьютерного зрения

В зависимости от вида решаемой задачи конкретизируется наполнение каждого блока представленного контейнера. В рамках дипломной работы в первую очередь решается задача – оценка позы человека, в частности анализ позы руки. Вторично решается задача классификации жеста на основе ключевых точек. На основе обобщенной схемы (рис. 1) была составлена схема, решающая задачи проекта (рис. 2).

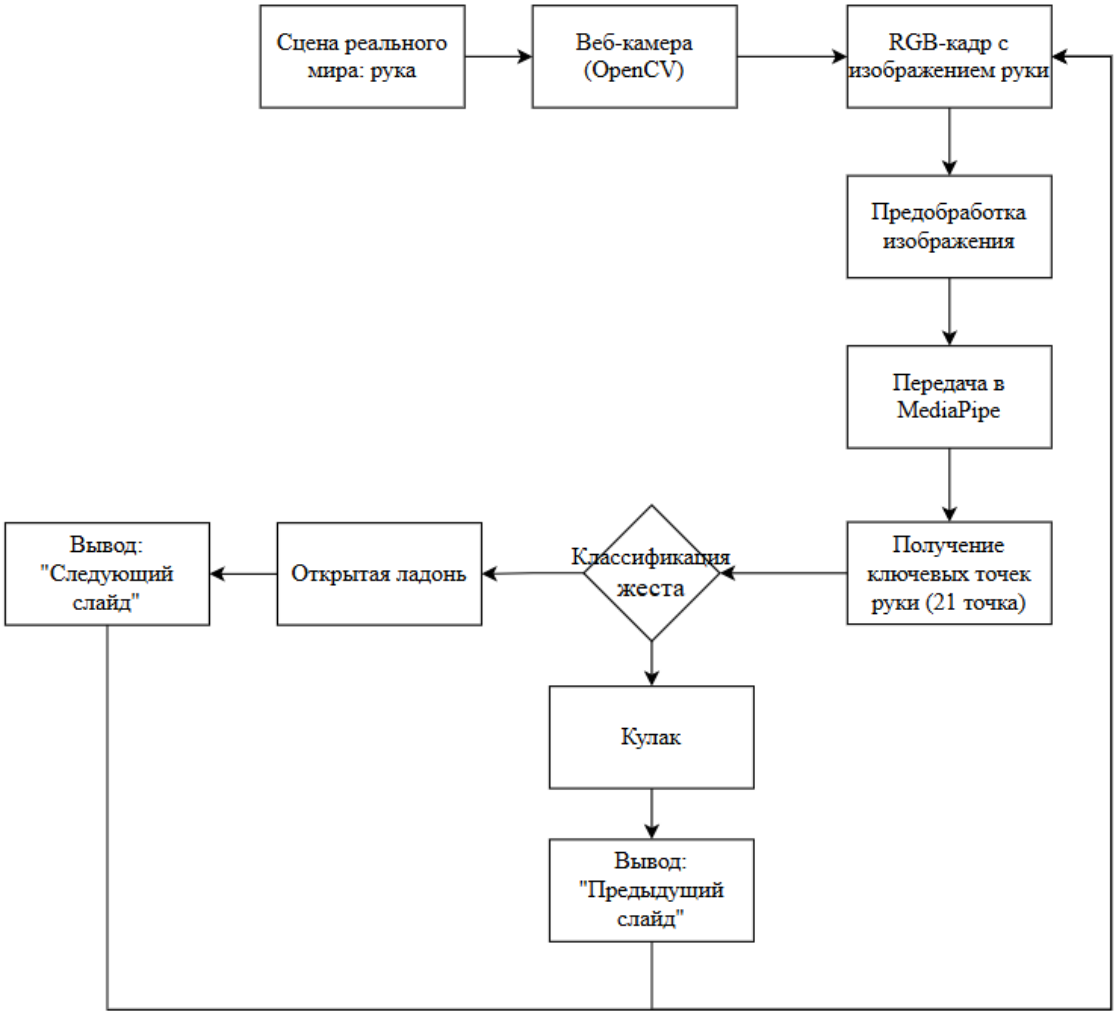


Рисунок 2 – Схема решения задачи оценки позы руки + классификация

1.2 Сравнительный анализ библиотек для распознавания жестов

Для написания дипломной работы был выбран язык программирования Python, совмещающий в себе баланс простоты, мощности. Он обладает интуитивно понятным синтаксисом, в нем много встроенных библиотек, он поддерживает принципы ООП, а также Python – один из лучших языков для работы с AI, NLP, CV.

В данной части работы представлен краткий обзор и анализ библиотек на языке Python, используя официальную документацию фреймворков, а также источники сети интернет.

MediaPipe (от Google) [12] – используется для обнаружения и отслеживания рук, лица, тела, жестов. Преимущества заключаются в том, что

в библиотеке уже представлены готовые модели для распознавания жестов, к тому же очень точно и быстро определяются положения руки. Работает на CPU и GPU, поддерживает мобильные устройства, что добавит расширяемости проекту. Также данный фреймворк легко интегрируется с OpenCV. Из недостатков выделяют то, что нельзя дообучить под свои жесты, имеются только предобученные модели.

OpenCV (Open Source Computer Vision Library) [14] – используется для компьютерного зрения, включая обработку изображений и видео, а также обнаружение рук и трекинг движений. Считается довольно универсальной библиотекой для обработки изображений, в ней можно реализовать собственную логику распознавания жестов (например, на основе контуров, цвета кожи, движения). Отлично подходит для нестандартных задач или кастомных решений, но в тоже время требует больше ручной работы и настройки.

TensorFlow / PyTorch [17][20] – используется для построения и обучения собственных нейросетей для классификации жестов. В случае данной библиотеки возможностей для кастомных моделей под любые жесты, но требуется разметка и подготовка определенного набора данных (например, ASL – язык жестов). Соответственно, сложнее реализация по сравнению с первыми двумя вариантами. Также библиотека поддерживает CNN, RNN, Transformers.

Handtrack.js (через браузер или WebView) [10] – библиотека подходит, если используется Python с веб-интерфейсом, можно встраивать JavaScript-библиотеки через Flask/Django.

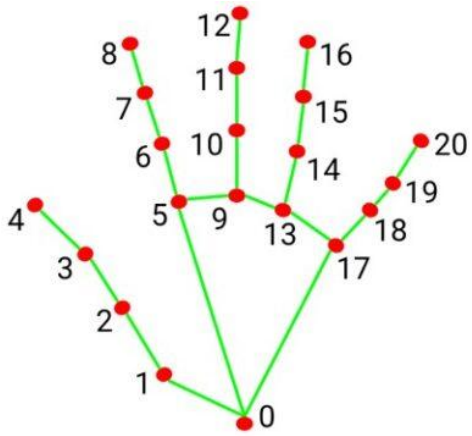
Дипломный проект заточен на распознавание жестов для управления презентацией, поэтому оптимальным вариантом, совмещающим простоту, распознавание в реальном времени, несложные настройки и готовые модели, стали библиотеки MediaPipe + OpenCV. Они позволяют реализовать точное и

быстрое распознавание жестов с минимальными затратами вычислительных ресурсов.

1.3 Модели кистей и определение жестов

Для начала необходимо определить функционал, который будет доступен пользователю для жестового управления. В случае управления презентации можно выделить следующие функции: слайд «вперед» и «назад». В итоге следует задача описать жесты для поддержания заданной функциональности. В данном случае два противоположных действия, поэтому можно выбрать два диаметрально противоположных жеста: вперед – раскрытая ладонь, назад – кулак.

Кисть руки представляется моделью, состоящей из набора точек, каждая из которых представляет один из суставов кисти (рис. 3). Изображение взято с сайта с официальной документацией.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Рисунок 3 – Модель руки

Важно отметить, что каждая точка модели представляется вектором координат (x, y, z), где координата z отвечает за «глубину» точки» [4].

Согласно официальной документации MediaPipe, в библиотеке предобучена модель Hand Landmarker [7], которая использует комплект моделей, состоящий из двух моделей: модель обнаружения ладоней и модель определения ключевых точек руки. Модель определения ключевых точек руки определяет расположение 21 точек, координаты сустава руки, в

обнаруженных областях руки. Модель была обучена примерно на 30 тысячах реальных изображений, а также на нескольких визуализированных синтетических моделях рук, наложенных на различный фон. А модель обнаружение ладони находит руки на входном изображении.

Поскольку обнаружение ладони требует длительного времени и вычислительных затрат, в режиме видео или потока Hand Landmarker использует ограничивающий прямоугольник (bounding box) из предыдущего кадра для локализации руки в следующих кадрах.

Hand Landmarker повторно запускает модель обнаружения ладоней только в том случае, если модель ключевых точек перестает находить руку или не может отслеживать ее в кадре. Это уменьшает количество раз, когда Hand Landmarker активирует модель обнаружения ладоней и ускоряет работу.

1.4 Сравнительный анализ библиотек для работы с презентацией

Для управления презентацией есть несколько вариантов: эмуляция клавиш (вперед, назад, стрелочки и т.п.), различные API. В данном разделе будут рассмотрены соответствующие фреймворки.

Следует уточнить, что API (Application Programming Interface) – программный интерфейс приложения, набор инструкций, который позволяет разным приложениям общаться между собой. API позволяет пользоваться уже разработанным функциям, а не писать их с нуля.

Самый универсальный подход в рамках работы – это эмуляция нажатий клавиш. Он работает с любой программой, а также с презентацией любого расширения (например, PowerPoint, PDF). Для этого в Python используется библиотека PyAutoGUI [15]. Она легка в установке и использовании. Из неоднозначного можно отметить, что она не работает с программой напрямую, как API, а лишь имитирует поведение пользователя – нажатие клавиш и перемещение курсора. Это означает, что библиотека не

может получать данные о текущем состоянии презентации, например, номер слайда или название файла. Несмотря на это, pyautogui остаётся популярным выбором за счёт своей кроссплатформенности и простоты.

Другой способ управления – через API PowerPoint, предоставляемый Microsoft. Для доступа к нему применяется библиотека pywin32 [18], позволяющая взаимодействовать с PowerPoint на уровне COM-интерфейса.

COM (Component Object Model) [13] – это независимая от платформы распределённая объектно-ориентированная система для создания двоичных компонентов программного обеспечения, которые могут взаимодействовать. Проще говоря, это технология от Microsoft, которая позволяет одним программам управлять другими через специально описанные интерфейсы. Процесс происходит не через имитацию действий пользователя, а напрямую – на уровне объектов и методов. Благодаря этому можно, например, запустить PowerPoint, открыть нужную презентацию, перейти к конкретному слайду и получить данные о текущем показе. Взаимодействие с COM в Python осуществляется через библиотеку pywin32, которая открывает доступ к методам и свойствам приложений Microsoft Office. В данном проекте это даёт возможность точного управления слайдами PowerPoint, а не просто эмуляции клавиш, как в случае с pyautogui.

Для управления Google-презентациями в облаке предусмотрено Google Slides API [8]. Оно подключается через библиотеку google-api-python-client и требует настройки учётных данных. С его помощью можно создавать и редактировать презентации, добавлять слайды и элементы на них, а также считывать структуру документа. Однако переключение слайдов в режиме показа недоступно – API не поддерживает управление презентацией в реальном времени. Вследствие этого в ряде задач Google Slides API используется совместно с инструментами автоматизации браузера, такими как Selenium или pyautogui.

Проанализировав описанные выше библиотеки, был сделан выбор в пользу эмуляции клавиш для большей универсальности и расширяемости, но с интеграцией PowerPoint API для корректной работы pptx-презентациями.

1.5 GUI (Graphical user interface) – графический интерфейс пользователя

Графический интерфейс пользователя – это форма взаимодействия пользователя с электронными устройствами через графические элементы: иконки, визуальные индикаторы и вспомогательные обозначения[1]. В отличие от текстовых интерфейсов (CLI), где управление осуществляется с помощью командной строки, GUI предлагает более интуитивный способ работы, устраняя необходимость запоминания сложных текстовых инструкций.

К преимуществам графического интерфейса относится наличие более дружелюбной (с англ. Friendly-user) системы управления ОС по сравнению со стандартной консолью. Также возможность разбивать настройки по группам для удобства администрирования.

К недостаткам GUI относят повышенное потребление системных ресурсов, особенно оперативной памяти. Это связано с тем, что все графические объекты, используемые в графическом интерфейсе загружены в оперативную память на постоянной основе.

1.6 Сравнительный анализ библиотек для реализации GUI

В данной части работы представлен краткий обзор и анализ GUI Framework на языке Python, используя официальную документацию фреймворков, а также источники сети интернет [23][24].

Tkinter[21] – стандартный и надежный инструмент для работы с графическим интерфейсом на языке программирования Python GUI. Благодаря простому использованию Tkinter является наиболее часто используемым каркасом GUI в Python.

Преимущество Tkinter в том, что он включен в стандартную библиотеку Python. Благодаря простому обращению часто используется в связке с «глубоким обучением». Для удобства расположения виджетов Tkinter дает путь к геометрическому расположению, благодаря чему можно разместить виджеты в любом месте и порядке.

Из недостатков хотелось бы отметить устаревший дизайн, который не соответствует современным UI-трендам, ограниченная функциональность для сложных интерфейсов.

PyQt[16] – это мощный кроссплатформенный фреймворк на основе Qt.

Из преимуществ отмечают: промышленный стандарт для профессиональных приложений, гибкость и высокая производительность.

Недостатком является то, что при сохранении файл имеет расширение .ui. В дальнейшем придется преобразовывать в файл с расширением .py. Также фреймворк сложен для освоения новичками и может потребоваться коммерческая лицензия.

Фреймворк Kivy GUI [11] – библиотека Python с открытым исходным кодом, используемая для создания нового пользовательского интерфейса и является идеальным выбором в разработке мобильных приложений, игр и интерактивных панелей. Но в то же время один код можно использовать для всех платформ. Фреймворк не использует нативные виджеты, что выглядит нетипично для ОС.

Еще один из фреймворков Python GUI – WxPython[22]. Это кроссплатформенный графический пользовательский интерфейс на языке программирования Python. WxPython имеет открытый исходный код, благодаря чему можно бесплатно использовать в своих целях. Также фреймворк отмечается хорошей производительностью и использованием нативых виджетов.

Статистика популярности произведена с помощью сервиса Google Trends [9] и представлена на рисунке 1.

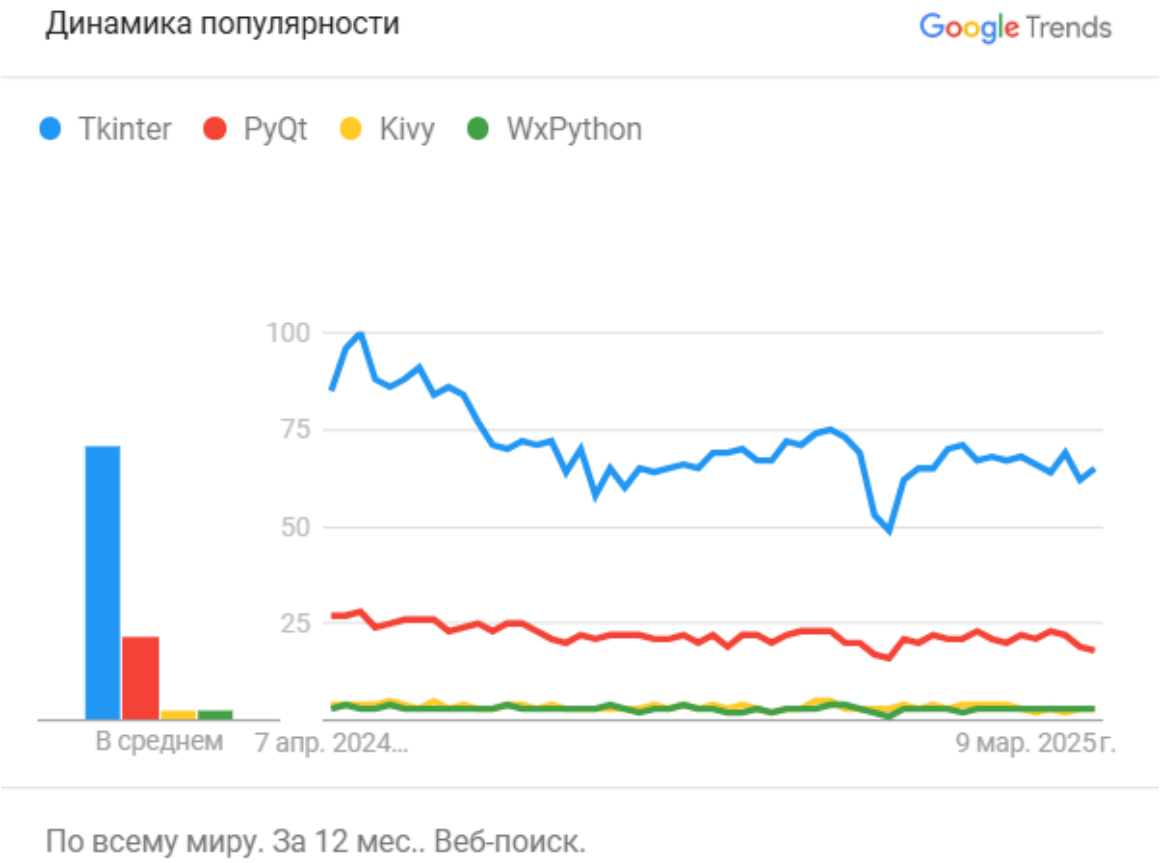


Рисунок 4 – График популярности за последний год

На основе анализа был выбран для работы проверенный фреймворк Tkinter.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО МОДУЛЯ

2.1 Требования к системе

Для начала работы хотелось бы выделить требования к системе. Требования разделяются на несколько уровней: уровень бизнес-требований, уровень пользовательских требований и уровень продуктных требований.

Святослав Куликов в своей книге «Тестирование программного обеспечения» подробно описывает, какое требование за что отвечает:

«Бизнес-требования выражают цель, ради которой разрабатывается продукт.

Пользовательские требования описывают задачи, которые пользователь может выполнять с помощью разрабатываемой системы. Пользовательские требования оформляются в виде пользовательских сценариев.

Функциональные требования описывают поведение системы.

Нефункциональные требования описывают свойства системы (удобство использования, безопасность, надежность, расширяемость и т.д.), которыми она должна обладать при реализации своего поведения.

Ограничения представляют собой факторы, ограничивающие выбор способов и средств реализации продукта».

Для выпускной квалификационной работы были выделены следующие требования, описанные в таблице 1.

Таблица 1 – Требования к программному продукту

Тип требования	Состав требований
Бизнес-требование	• Нужен инструмент, позволяющий в режиме реального времени бесконтактно управлять презентацией с помощью жестов. • Снизить зависимость от дополнительного оборудования (например, пульты, кликеры)

	за счет использования веб-камеры.
Функциональные требования	• Распознавание руки через библиотеку MediaPipe. • Отслеживание жестов и автоматическая отправка клавиш → и ← в PowerPoint. • Отображение статуса программы в отдельном плавающем окне. • Калибровка камеры и микрофона перед использованием. • Настройка задержки между жестами (1–30 сек) для предотвращения случайных срабатываний.
Нефункциональные требования	• Производительность: ○ задержка между жестом и действием – не более 1 сек; ○ работа с разрешением камеры от 640x480 при 30 FPS. • Удобство использования (UX): ○ все кнопки подписаны; ○ минималистичный дизайн с кнопками старта/остановки и настройками; ○ по умолчанию размер окна, на котором все видно. • Совместимость: ○ операционная система – Window 7/10/11 (поддержка Python и Tkinter); ○ PowerPoint 2016-2023;

	○ веб-камеры и микрофоны, поддерживаемые OpenCV. • Надежность: ○ при сбое выводится понятное сообщение.
Ограничения	• Требуется активное окно PowerPoint в режиме презентации. • Система рассчитана на одного пользователя (одну руку в кадре).

2.2 Пользовательские требования

Для оформления пользовательских требований выбрали два способа:

1. User Story – короткие сценарии от лица пользователя;
2. Диаграмма Use Case (рис. 5).

Для сценариев использования были выделены следующие категории пользователей:

- Преподаватели – используют систему для более удобного управления презентациями во время лекций.
- Лекторы и докладчики – выступают на конференциях и используют систему для бесконтактного управления слайдами.
- Пользователи с ограниченными возможностями – нуждаются в бесконтактном способе управления презентацией, который бы заменил стандартные устройства ввода.

Проанализировав предметную область и другие требования, были написаны следующие User Story:

1. Как преподаватель, я хочу управлять слайдами жестами, чтобы не отвлекаться от лекции и сохранить внимание студентов.
2. Как преподаватель, я хочу иметь возможность быстро пролистывать слайды назад жестом, чтобы отвечать на вопросы студентов и показывать предыдущий материал.
3. Как преподаватель, я хочу использовать систему жестов без необходимости настройки перед каждой лекцией, чтобы экономить время.
4. Как докладчик на конференции, я хочу переключать слайды жестами, чтобы удерживать контакт с аудиторией и делать презентацию более живой.
5. Как докладчик, я хочу получать визуальное подтверждение успешного распознавания жеста, чтобы быть уверенным в смене слайда.
6. Как пользователь с ограниченными возможностями, я хочу управлять презентацией жестами, чтобы иметь возможность выступать без традиционных устройств управления.
7. Как пользователь с ограниченными возможностями, я хочу настраивать время между жестами под себя, чтобы удобнее взаимодействовать с системой.
8. Как пользователь с ограниченными возможностями, я хочу получать звуковое или визуальное подтверждение распознавания жестов, чтобы знать, что команда выполнена.

Рисунок 5 – Диаграмма Use Case

ГЛАВА 3. РАЗРАБОТКА КОМПОНЕНТОВ ПРОГРАММНОГО МОДУЛЯ

3.1. Настройка окружения для работы

В ходе теоретической части работы был проанализирован и выбран набор специализированных библиотек. Для настройки окружения они были подключены:

1. OpenCV (cv2) – это основной инструмент для работы с видеопотоком, отвечающий за захват изображения с веб-камеры и предварительную обработку кадров.
2. MediaPipe – это главная библиотека для распознавания жестов.
3. PyAutoGUI – обеспечивает интеграцию с системой презентации, эмулирует нажатия клавиш.
4. tkinter и его подмодули – это основная библиотека для создания графического интерфейса. С помощью него реализованы все страницы приложения: главное меню, окно калибровки, окно с оверлеем.
5. Pillow – основная задача библиотеки в рамках дипломной работы – конвертация кадров из OpenCV в формат tkinter
6. Win32com.client – библиотека для интеграции с PowerPoint.
7. SpeechRecognition – библиотека для обработки аудиопотока.
8. Threading – стандартная библиотека для реализации многопоточности: параллельной обработки видео и аудио, фоновой работы голосового модуля.
9. Time – стандартная библиотека для работы с временными интервалами.

Для успешной работы приложения все зависимости были перенесены в файл requirements.txt.

3.2 Реализация базового функционала приложения

Базовая функциональность приложения заключается в распознание жестов пользователя, обозначающих слайд вперед/слайд назад, и

соответственное управление презентацией PowerPoint. В рамках реализации базового функционала можно выделить три подзадачи: отслеживание положения руки с помощью камеры, интерпретация определенных жестов (жесты были выбраны в 1.3) и эмуляция нажатия клавиш Right/Left для переключения между слайдами.

Рассмотрим каждую подзадачу подробнее. Для отслеживания руки используется библиотека MediaPipe с предобученной моделью Hands, которая в режиме реального времени определяет положение кисти и пальцев. Инициализация и создание экземпляра модели, а также захват видео с помощью cv2, представлен на рисунке 7.

```
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils

hands = mp_hands.Hands(max_num_hands=1, min_detection_confidence=0.7, model_complexity=1)
cap = cv2.VideoCapture(0)
```

Рисунок 7 – Листинг с инициализацией MediaPipe

Далее на рисунке 8 представлена функция обработки видеокадров, в которой считывается кадр с веб-камеры, обнаруживает руку, определяет жесты и выполняет соответствующее действие в презентации.

```
def process_frame():
    global last_gesture_time
    ret, frame = cap.read()
    if not ret:
        return
    frame = cv2.flip(frame, 1)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    results = hands.process(rgb)
    now = time.time()

    if results.multi_hand_landmarks and results.multi_handedness:
        for idx, hand_landmarks in enumerate(results.multi_hand_landmarks):
            handedness = results.multi_handedness[idx].classification[0].label
            fingers = count_extended_fingers(hand_landmarks, handedness)

            # Жест "ВПЕРЁД" (все пальцы разогнуты ✋)
            if fingers == [1, 1, 1, 1, 1] and now - last_gesture_time >= gesture_delay:

                pyautogui.press('right')  # Листает слайд вперёд
                last_gesture_time = now
                print("Жест: ВПЕРЁД ✋")

            # Жест "НАЗАД" (все пальцы сжаты ✊)
            elif fingers == [0, 0, 0, 0, 0] and now - last_gesture_time >= gesture_delay:

                pyautogui.press('left')  # Листает слайд назад
                last_gesture_time = now
                print("Жест: НАЗАД ✊")
```

Рисунок 8 – Листинг функции обработки видеокадра

Также для наглядности был взять PowerPointAPI, точнее win32com. С помощью данной библиотеки была написана функция get_powerpoint_info(), в которой программа подключается к PowerPoint (создает соединение с запущенным экземпляром презентации), получает активную презентацию (т.е. презентацию, запущенную в режиме слайд-шоу) и возвращает значение текущего слайда и сколько слайдов представлено всего. Листинг этой функции представлен на рисунке 9.

```
def get_powerpoint_info():
    try:
        ppt = win32com.client.Dispatch("PowerPoint.Application")
        presentation = ppt.ActivePresentation
        total_slides = presentation.Slides.Count
        current_slide = presentation.SlideShowWindow.View.CurrentShowPosition
        return total_slides, current_slide
    except Exception as e:
        #print("PowerPoint ошибка:", e)
        return None, None
```

Рисунок 9 – Листинг функции с получением информации о текущей презентации

Также в этой функции представлена обработка ошибок на тот случай, если PowerPoint не запущен, презентация не открыта или не в режиме слайд-шоу, в таком случае просто не вернется количество слайдов.

3.3 Создание графического интерфейса

Основной функционал, заявленный в требованиях, реализован, но пользователю не будет удобно пользоваться таким приложением. В подобных приложениях крайне необходим графический интерфейс, подтверждающий, что приложение работает корректно, что отображен нужный слайд, и показывающий, что приложение доступно, какой жест был распознан последний.

Графический интерфейс был реализован с использованием библиотеки Tkinter и включает в себя несколько компонентов:

- главный приветственный экран с кнопками запуска, калибровки и настройки приложения представлен на рисунке 10;

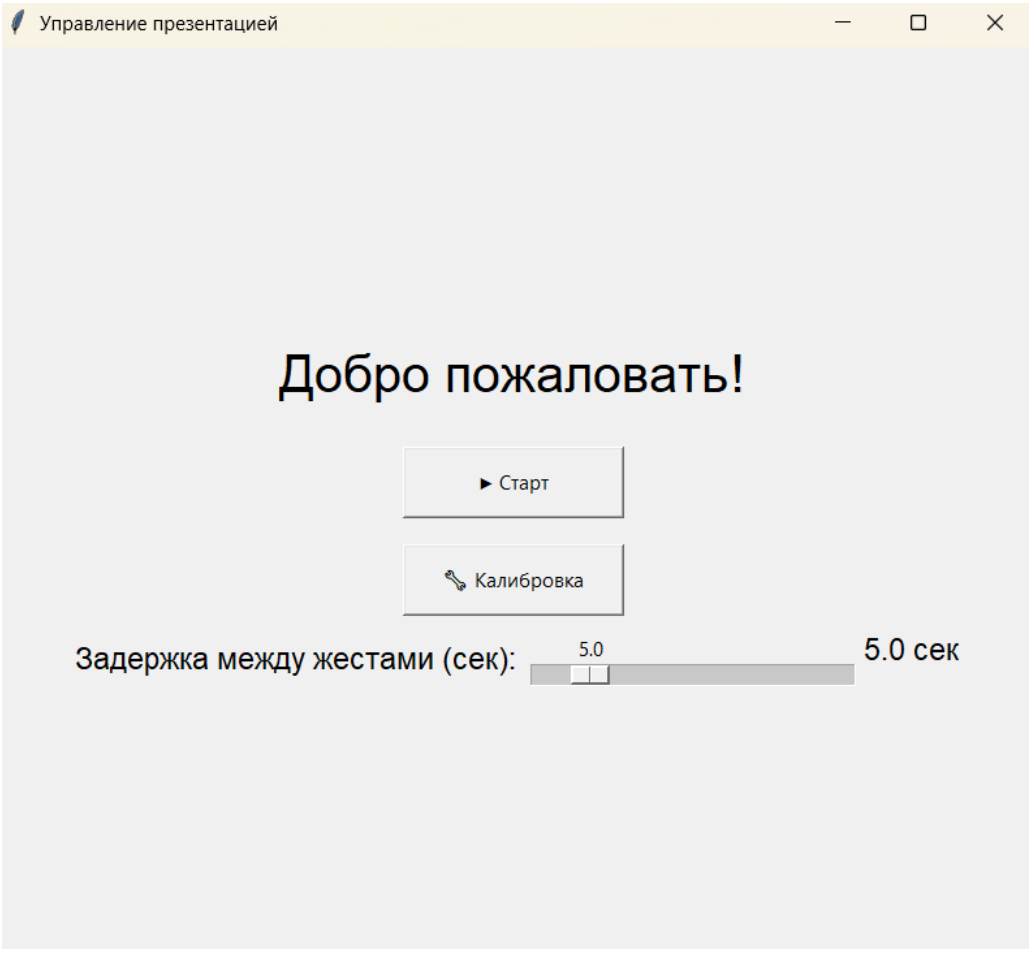


Рисунок 10 – Графический интерфейс главного окна

- окно, появляющееся после нажатия кнопки «старт», с отображением видео с камеры и текущего распознанного жеста представлено на рисунке 11;

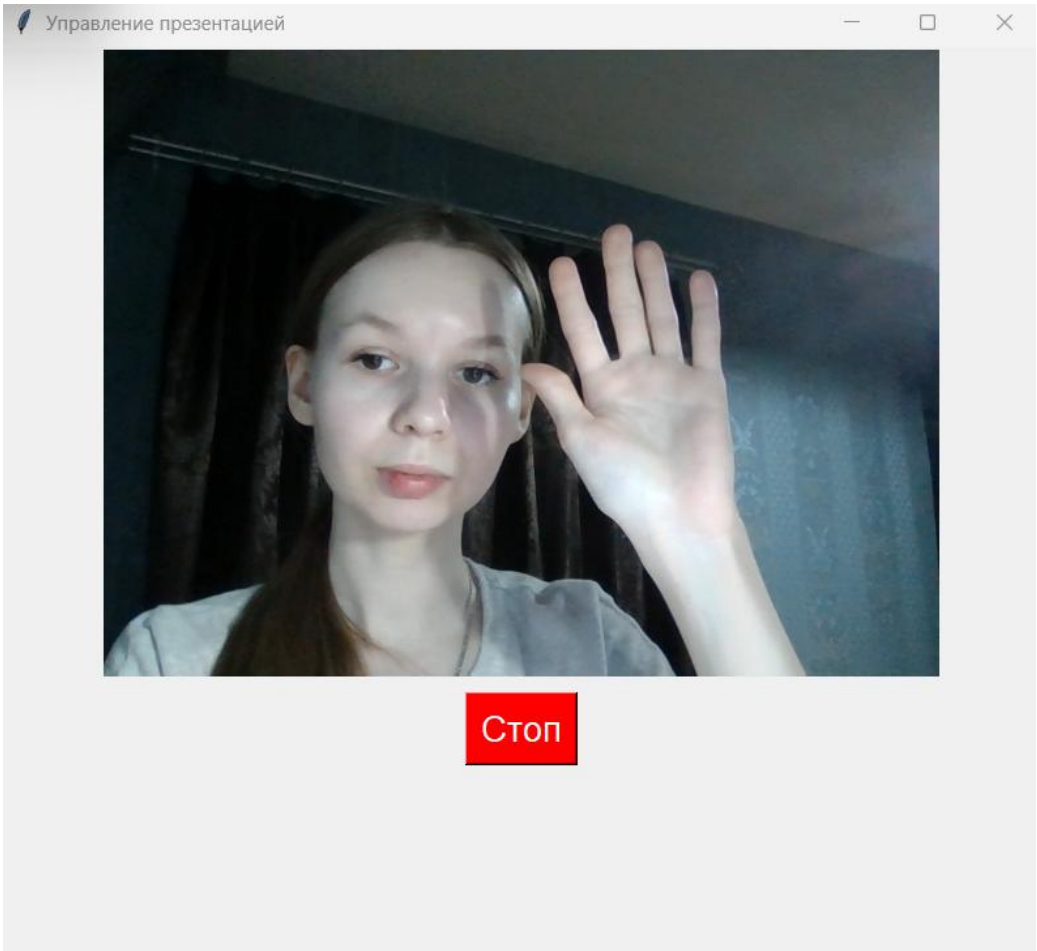


Рисунок 11 – Графический интерфейс окна управления презентацией

- оверлей-окно, которое находится всегда поверх остальных окон и отображает текущий статус системы и жест, представлено на рисунке 12;

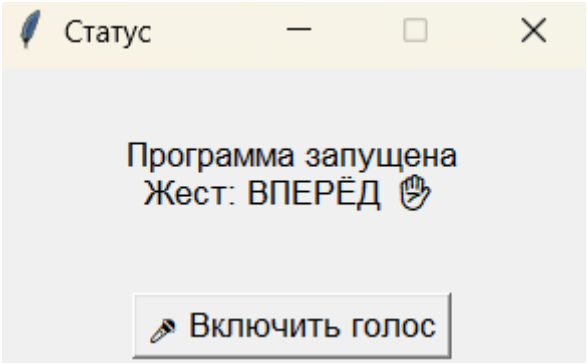


Рисунок 12 – Графический интерфейс оверлей-окна

- окно калибровки, в котором будет две вкладки:
  - вкладка «Камера» – позволяет выбрать и протестировать камеру (рис.13), отображает изображение с выделенными

линиями модели MediaPipe (рис.14) и уведомление при отсутствии руки более 10 секунд (рис.15);

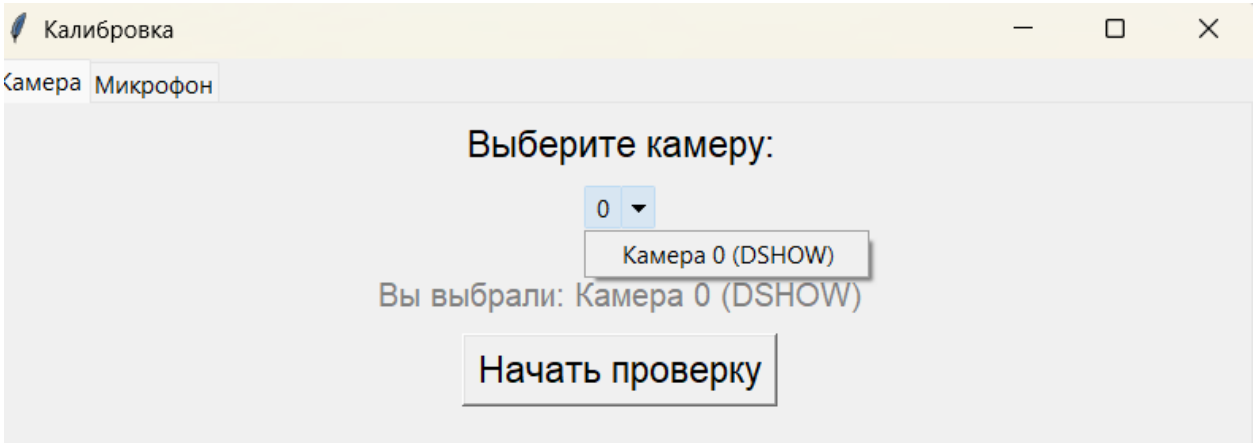


Рисунок 13 – Графический интерфейс окна калибровки вкладки «Камера»

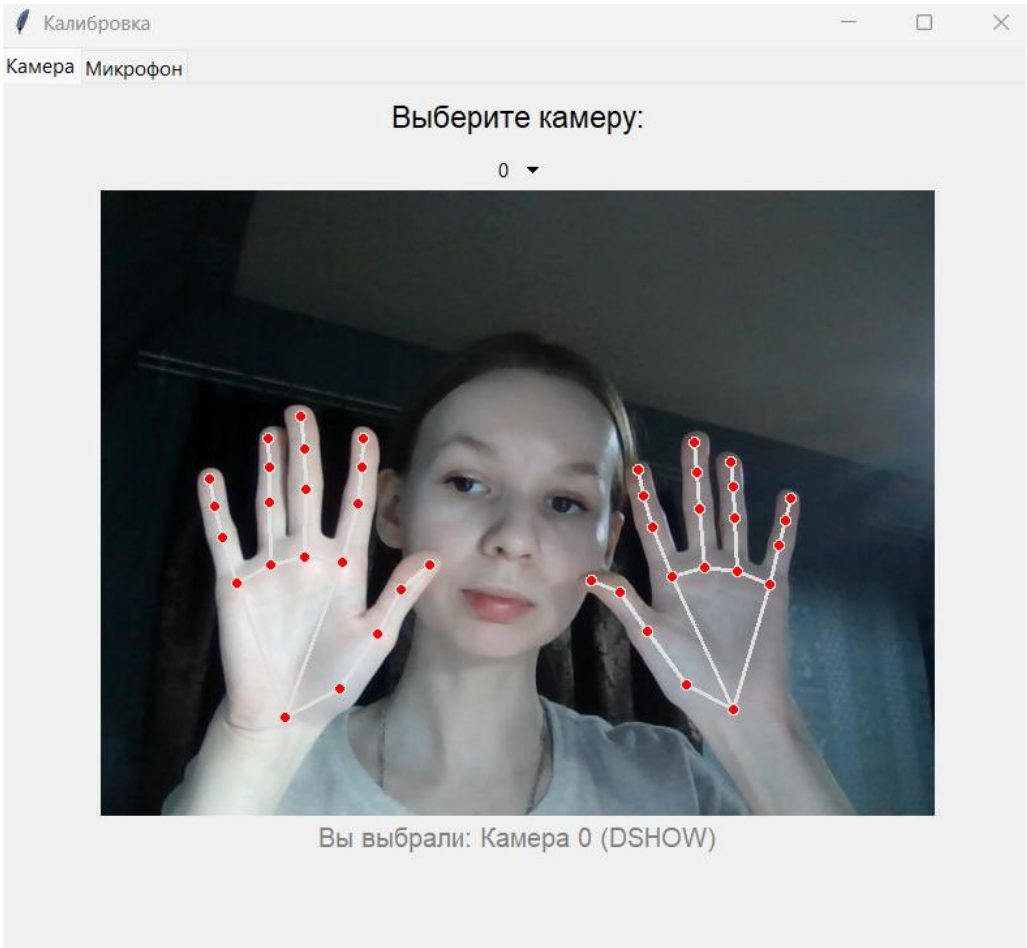


Рисунок 14 – Окно калибровки с изображением в реальном времени

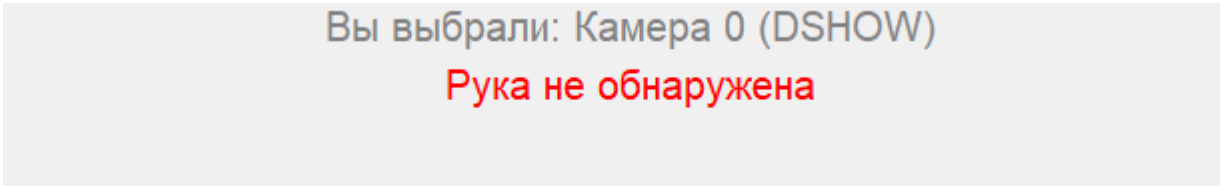


Рисунок 15 – Уведомление под изображением с предупреждением, что руки не обнаружены

- вкладка «Микрофон» – отображает список всех доступных аудиоустройств на компьютере с упрощенными названиями (рис.16), позволяя выбрать и проверить одно из них для голосового управления (рис.17);

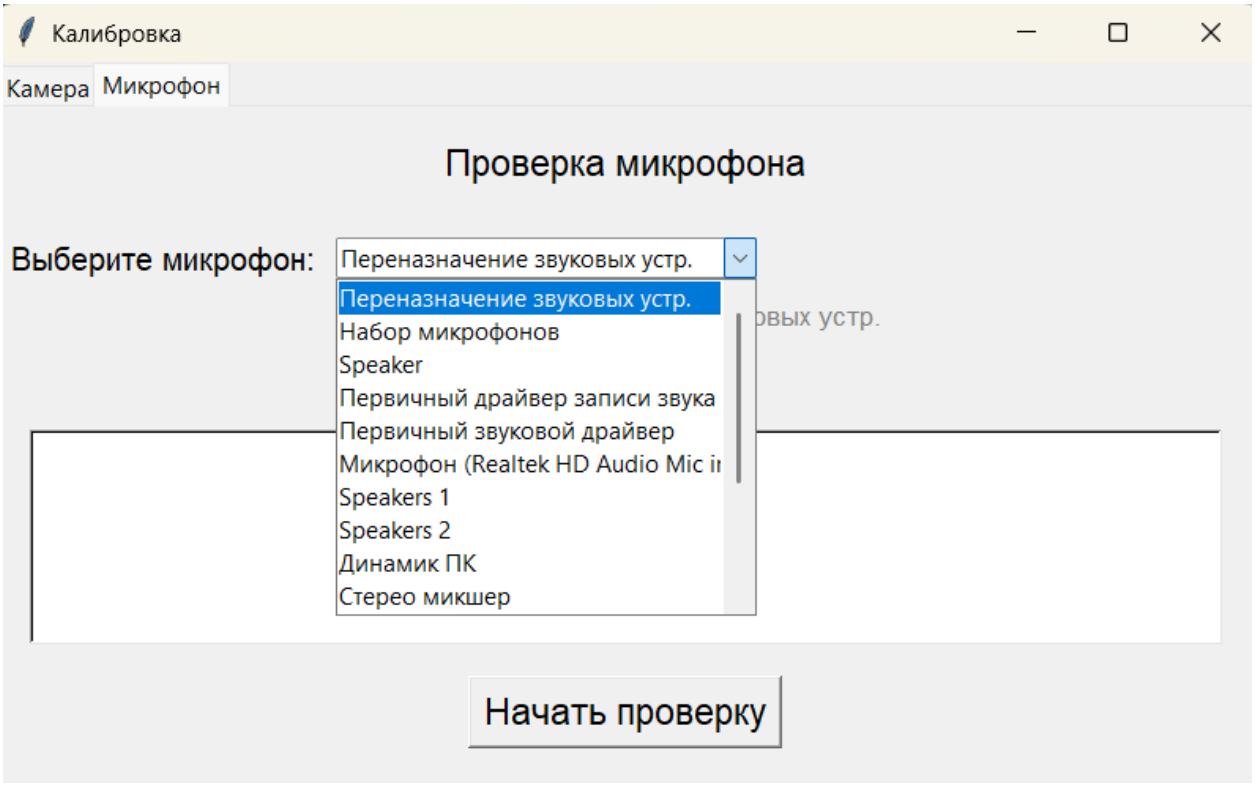


Рисунок 16 – Графический интерфейс окна калибровки вкладки «Микрофон»

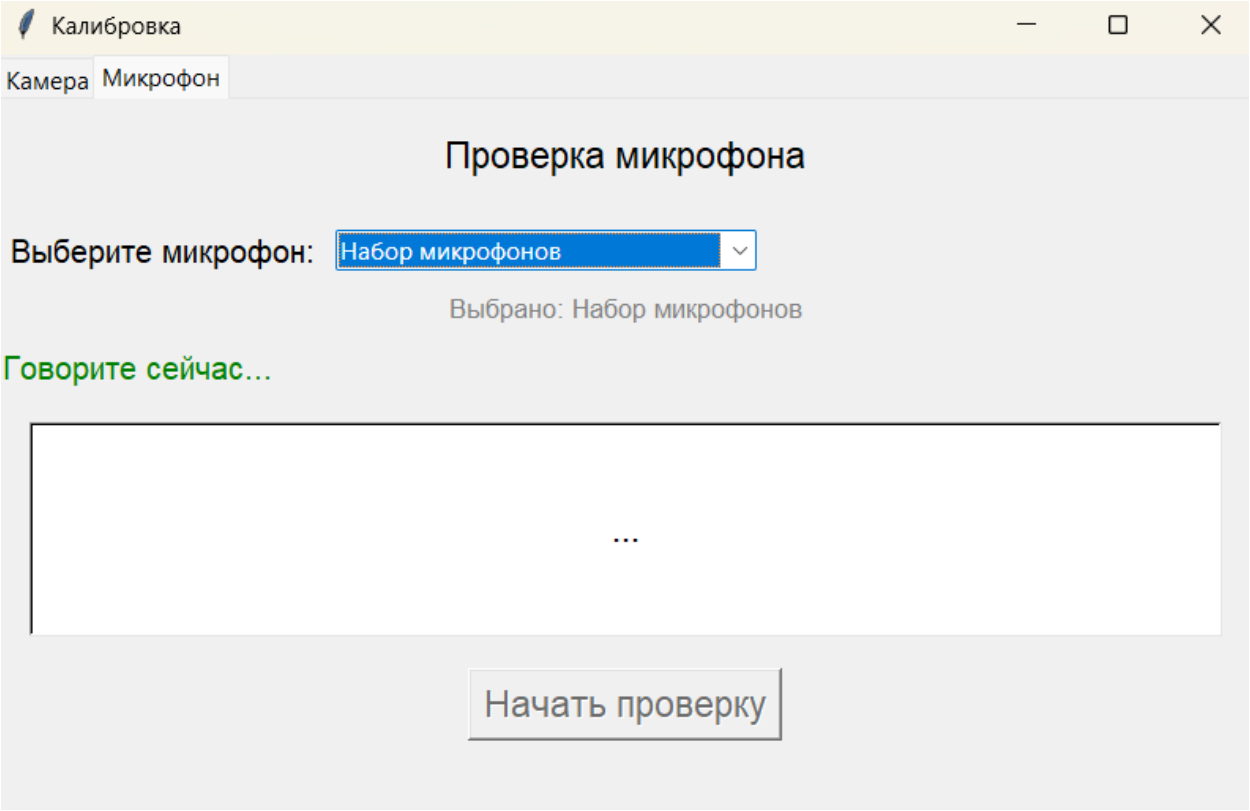


Рисунок 17 – Начата проверка распознавания голоса

- После того как началась проверка микрофона и распознания текста, который говорится в микрофон, есть несколько вариантов, что произойдет. Позитивный исход – Речь распознана и отображена в соответствующем окне (рис. 18). Негативный исход – микрофон не найден, время на распознание вышло, аудиоустройство некорректно (например, выбраны колонки, которые не имеют микрофона), а также может быть ошибка микрофона (рис. 19, 20).

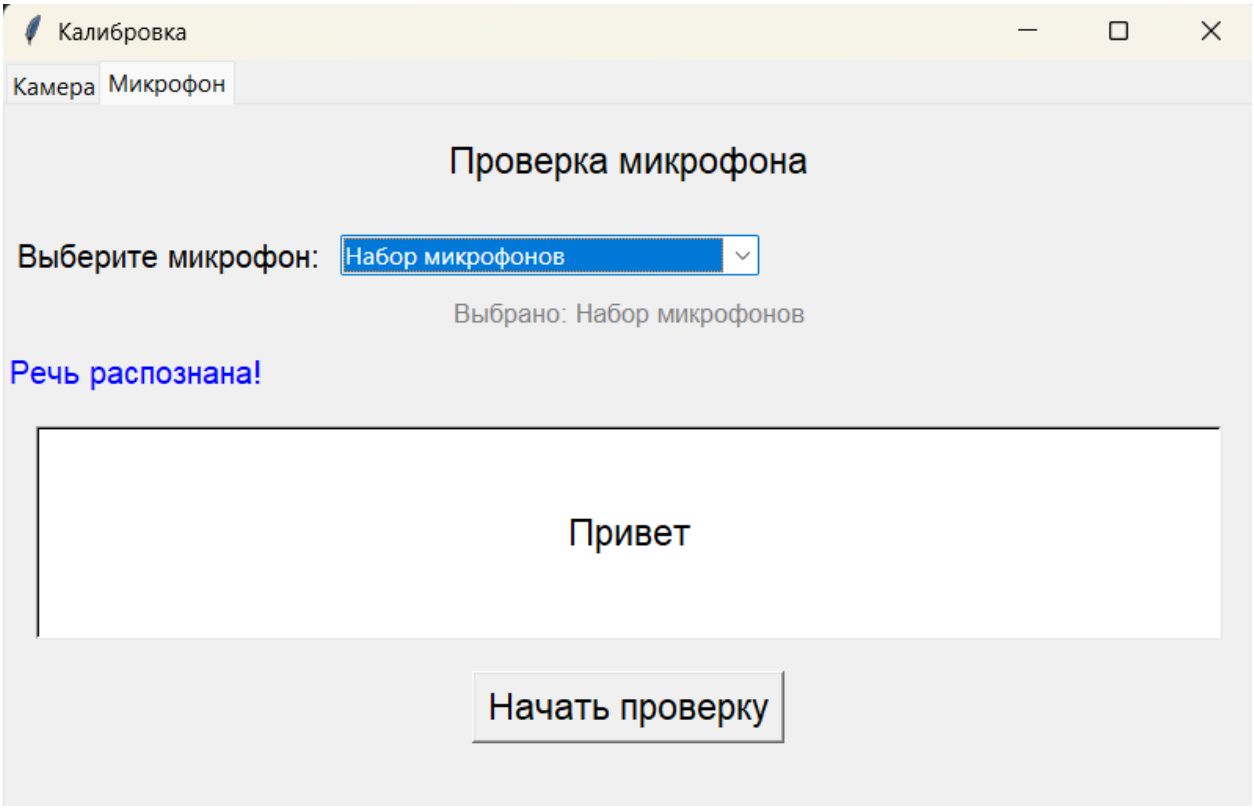


Рисунок 18 – Позитивный исход проверки

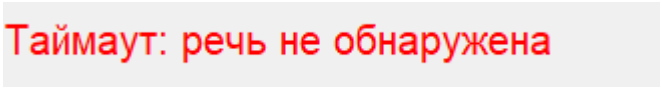


Рисунок 19 – Негативный исход проверки

Рисунок 20 – Негативный исход проверки

Графические элементы удобно расположены и сгруппированы. Отображение видео в окне осуществляется через PIL.Image и ImageTk.PhotoImage. Настройка параметров на рисунке 10 осуществляется с помощью ползунка.

3.4 Калибровка

В разделе 3.3 была упомянута калибровка, рассмотрим ее реализацию подробнее.

Калибровка – важный этап, без которого точность распознавания жестов и голосовых команд была бы значительно ниже. В рамках выпускной квалификационной работы под калибровкой подразумевается процедура настройки камеры и микрофона, а также проверка корректности захвата руки и распознавания речи в режиме реального времени.

Главная цель калибровки – предоставить пользователю возможность выбрать устройство ввода (веб-камеру и микрофон), а также проконтролировать, как система отображает ключевые точки на руке и насколько точно воспринимает голосовые команды. Благодаря этому можно убедиться, что рука правильно освещена, не выходит за границы кадра, а микрофон захватывает речь без существенных искажений.

Как говорилось ранее, в приложении имеется две вкладки. Во вкладке «Камера» реализована функция определения доступных устройств видеозахвата. Последовательно проверяются индексы от 0 до 4, и при успешной инициализации устройства в список добавляется его системное имя (например, «Камера 0 (MSMF)»). Результаты отображаются в выпадающем списке, позволяющем выбрать нужную камеру (рис.13).

После нажатия кнопки «Начать проверку» запускается трансляция с выбранной камеры и анализом видеопотока в режиме реального времени. Листинг функции этого процесса представлен на рисунке 21.

```
def show_camera_feed():
    show_cam_btn.pack_forget()
    temp_cap = cv2.VideoCapture(camera_var.get())
    hands_calib = mp_hands.Hands(min_detection_confidence=0.7, model_complexity=1)
    last_seen_hand = [time.time()]
    message_var = tk.StringVar()
    message_label = tk.Label(tab_camera, textvariable=message_var, font=("Arial", 12), fg="red")
    message_label.pack()

    def update():
        if not running_calib[0]:
            temp_cap.release()
            hands_calib.close()
            return

        ret, frame = temp_cap.read()
        if ret:
            frame = cv2.flip(frame, 1)
            rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            results = hands_calib.process(rgb)

            if results.multi_hand_landmarks:
                last_seen_hand[0] = time.time()
                for hand_landmarks in results.multi_hand_landmarks:
                    mp_drawing.draw_landmarks(frame, hand_landmarks, mp_hands.HAND_CONNECTIONS)
                message_var.set("")
            else:
                if time.time() - last_seen_hand[0] > 10:
                    message_var.set("Рука не обнаружена")

            img = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
            imgtk = ImageTk.PhotoImage(image=img)
            video_label_calib.imgtk = imgtk
            video_label_calib.configure(image=imgtk)

        tab_camera.after(10, update)

    update()
```

Рисунок 21 – Листинг функции show_camera_feed()

Во второй вкладке было решено придерживаться похожего алгоритма, что и во вкладке с камерой: выпадающий список с аудиоустройствами, но с упрощенными отображаемыми названиями и удалением дубликатов. При этом сохраняется связь между упрощёнными и оригинальными названиями, чтобы при запуске распознавания использовать точное имя устройства. Листинг функции этого процесса представлен на рисунке 22.

```
mic_list = sr.Microphone.list_microphone_names()
simplified_mic_list = []

for mic in mic_list:

    simplified = mic.split('(')[0].strip()
    simplified = simplified.split('-')[0].strip()
    simplified = simplified.split('|')[0].strip()
    simplified = simplified.replace("Microphone", "").replace("Микрофон", "").strip()

    # Если после упрощения название пустое, оставляем оригинальное
    if not simplified:
        simplified = mic

    if simplified not in simplified_mic_list:
        simplified_mic_list.append(simplified)

mic_mapping = {simplified: original for simplified, original in zip(simplified_mic_list, mic_list)}
```

Рисунок 22 – Листинг кода с формированием списка аудиоустройств

Аналогично, после нажатия кнопки «Начать проверку» создается отдельный поток, в котором производится запись речи через выбранное устройство. Используется автоматическая калибровка по уровню окружающего шума, а затем происходит захват и распознавание речи с ограничением по времени. Текст распознанной речи выводится на экран, а также обновляется статус: успешно ли распознано, не было ли таймаута или ошибки. Код этой функции представлен на рисунке 23.

```
def start_mic_test():
    nonlocal mic_recording
    if not simplified_mic_list:
        return

    start_mic_btn.config(state="disabled")
    mic_status_label.config(text="Говорите сейчас...", fg="green")
    mic_text_label.config(text="...")
    mic_recording = True

    selected_mic_name = mic_mapping.get(mic_var.get(), mic_list[0])
    selected_mic_index = mic_list.index(selected_mic_name)

    def listen_thread():
        nonlocal mic_recording
        try:
            with sr.Microphone(device_index=selected_mic_index) as source:
                recognizer.adjust_for_ambient_noise(source, duration=1)
                try:
                    audio = recognizer.listen(source, timeout=5, phrase_time_limit=5)
                    text = recognizer.recognize_google(audio, language="ru-RU")
                    mic_text_label.config(text=text)
                    mic_status_label.config(text="Речь распознана!", fg="blue")
                except sr.WaitTimeoutError:
                    mic_status_label.config(text="Таймаут: речь не обнаружена", fg="red")
                except sr.UnknownValueError:
                    mic_status_label.config(text="Речь не распознана", fg="red")
                except Exception as e:
                    mic_status_label.config(text=f"Ошибка: {str(e)}", fg="red")
        except Exception as e:
            mic_status_label.config(text=f"Ошибка микрофона: {str(e)}", fg="red")
        finally:
            mic_recording = False
            start_mic_btn.config(state="normal")

    threading.Thread(target=listen_thread, daemon=True).start()
```

Рисунок 23 – Листинг функции start_mic_test()

Единственный минус, который можно отметить при этом процессе, что необходимо стабильное подключение к сети Интернет для работы с распознаванием через Google API.

Выход из режима калибровки и закрытие окна калибровки останавливает поток и освобождает ресурсы камеры. Это позволяет избежать конфликтов с другими окнами системы или внешними приложениями, использующими видеопоток, а также защищает от возможных утечек памяти и конфликтов при повторных запусках.

3.5 Расширяемость приложения

Приложение поддерживает расширяемость за счёт интеграции голосового управления. Используется библиотека SpeechRecognition, позволяющая распознавать голосовые команды с микрофона. При включении голосового режима (через кнопку в оверлей-окне) запускается отдельный поток, обрабатывающий аудиоввод и интерпретирующий команды для управления презентацией аналогично жестам.

Для удобства используются однословные команды, что снижает риск ошибок при плохом качестве микрофона. Для переключения на следующий слайд было выбрано слово «вперед», а для переключения на предыдущий – слово «назад».

Программная реализация представлена на рисунке 24.

```python
def voice_control_loop():
    recognizer = sr.Recognizer()
    mic = sr.Microphone()
    global current_gesture_text

    with mic as source:
        recognizer.adjust_for_ambient_noise(source)

    while running and voice_enabled:
        try:
            with mic as source:
                print("🎤 Ожидание команды...")
                audio = recognizer.listen(source, timeout=5, phrase_time_limit=4)

            command = recognizer.recognize_google(audio, language="ru-RU").lower()
            print("Распознано:", command)

            gesture = None

            if "вперёд" in command:
                pyautogui.press('right')
                gesture = "Голос: ВПЕРЁД 🎤"
            elif "назад" in command:
                pyautogui.press('left')
                gesture = "Голос: НАЗАД 🎤"

            if gesture:
                ppt_total, ppt_current = get_powerpoint_info()
                if ppt_total and ppt_current:
                    current_gesture_text = f"{gesture}\nСлайд {ppt_current} из {ppt_total}"
                else:
                    current_gesture_text = gesture

        except sr.WaitTimeoutError:
            continue
        except sr.UnknownValueError:
            continue
        except sr.RequestError as e:
            print(f"Ошибка запроса к сервису распознавания: {e}")
            break
```

Рисунок 24 – Листинг кода функции с обработкой аудиопотока

Архитектура построена таким образом, чтобы компоненты (жестовое и голосовое управление) функционировали независимо и могли быть легко расширены.

Также расширяемость приложения достигается с использованием библиотеки для эмуляции клавиш (стрелочек). С помощью этой библиотеки функциональность приложения может быть использована не только в

PowerPoint презентациях, но и в презентациях формата PDF, GoogleSlides и необязательно в режиме слайд-шоу. Но в таком случае количество слайдов и контроль над текущим слайдом осуществляться не будет.

ЗАКЛЮЧЕНИЕ

При написании дипломной работы по теме исследования была реализована система управления презентацией PowerPoint с использованием компьютерного зрения, а также голосового распознавания. Особое внимание было уделено разработке графического интерфейса (GUI) на Python.

В теоретической части удалось полностью раскрыть базовые понятия, провести анализ и рассмотреть соответствующие библиотеки на языке программирования Python. На основе полученных материалов с помощью Visual Studio Code было смоделировано и реализовано приложение с интуитивно понятным интерфейсом, включающим:

- систему навигации между страницами;
- калибровку системы;
- начало работы с оверлей-окном;
- использование голосового управления по желанию пользователя.

В ходе выполнения работы были учтены вопросы удобства, надёжности и расширяемости приложения. Система построена модульно, что позволяет при необходимости внедрять дополнительные команды, совершенствовать алгоритмы распознавания или адаптировать интерфейс под иные сценарии использования.

Таким образом, можно сделать вывод о том, что поставленные задачи были выполнены, а цель – достигнута.

СПИСОК ЛИТЕРАТУРЫ

1. Ералева М. Е., Столбикова С. П., Мурзахматов М. А. Разработка GUI-интерфейса / Ералева М. Е., Столбикова С. П., Мурзахматов М. А. // Российские регионы в фокусе перемен: Сборник докладов XVIII Международной конференции, Екатеринбург, 16–18 ноября 2023 года. / Уральский федеральный университет имени первого Президента России Б.Н. Ельцина. – Екатеринбург: Издательский Дом «Ажур», 2023. – С. 594-599.

2. Кудрявцев Н. Г., Фролов И. Н. Практика применения компьютерного зрения и элементов машинного обучения в учебных проектах: учебное пособие / Кудрявцев Н. Г., Фролов И. Н. – Горно-Алтайск: ГАГУ, 2022. – 180 с. // Лань: электронно-библиотечная система [сайт]. – URL: https://e.lanbook.com/book/271100 (дата обращения: 12.05.2025). – Режим доступа: для авториз. пользователей.

3. Куликов С. С. Тестирование программного обеспечения. Базовый курс: производственно-практическое издание / Куликов С. С. – 3-е изд. – Минск: Четыре четверти, 2020. – 312 с.

4. Овчаров З. А., Новикова С. В. Метод распознавания жестов управления интерфейсом / Овчаров З. А., Новикова С. В. // Вестник Технологического университета: [электронная версия]. – 2023. – № 5. – С. 64-69. // Лань: электронно-библиотечная система [сайт]. – URL: https://e.lanbook.com/journal/issue/346658 (дата обращения: 14.05.2025). – Режим доступа: для авториз. пользователей.

5. Селянкин В. В. Компьютерное зрение. Анализ и обработка изображений: учебное пособие / Селянкин В. В. – 3-е изд., стер. – Санкт-Петербург: Лань, 2023. – 152 с. // Лань: электронно-библиотечная система [сайт]. – URL: https://e.lanbook.com/book/276455 (дата обращения: 17.05.2025). – Режим доступа: для авториз. пользователей.

6. 30 правил и секретов успешной презентации [Электронный ресурс] // Presium.pro: блог. – URL: https://presium.pro/blog/30-rules-and-

secrets-of-a-successful-presentation (дата обращения: 01.05.2025). – Режим доступа: свободный.

7. Google Developers. MediaPipe Hand Landmarker [Электронный ресурс] // Google AI for Developers. – URL: https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker?hl=ru (дата обращения: 15.04.2025). – Режим доступа: свободный

8. Google Slides API Documentation [Электронный ресурс]. – URL: https://developers.google.com/slides/api (дата обращения: 15.04.2025). – Режим доступа: свободный.

9. Google Trends [Электронный ресурс]. – URL: https://trends.google.ru/trends/explore?cat=31&q=Tkinter,PyQt,Kivy,WxPython&hl=ru (дата обращения: 09.04.2025). – Режим доступа: свободный.

10. Handtrack.js Documentation [Электронный ресурс]. – URL: https://github.com/victordibia/handtrack.js (дата обращения: 15.04.2025). – Режим доступа: свободный.

11. Kivy Documentation [Электронный ресурс]. – URL: https://kivy.org/doc/stable/ (дата обращения: 09.04.2025). – Режим доступа: свободный.

12. MediaPipe Documentation [Электронный ресурс]. – URL: https://developers.google.com/mediapipe (дата обращения: 15.04.2025) – Режим доступа: свободный.

13. Microsoft. Component Object Model (COM) [Электронный ресурс] // Microsoft Learn: документация. – URL: https://learn.microsoft.com/ru-ru/windows/win32/com/component-object-model--com--portal (дата обращения: 15.05.2025). – Режим доступа: свободный.

14. OpenCV Documentation [Электронный ресурс]. – URL: https://docs.opencv.org (дата обращения: 15.04.2025). – Режим доступа: свободный.

15. PyAutoGUI Documentation [Электронный ресурс]. – URL: https://pyautogui.readthedocs.io (дата обращения: 25.04.2025). – Режим доступа: свободный.

16. PyQt Documentation [Электронный ресурс]. – URL: https://www.riverbankcomputing.com/software/pyqt/intro/ (дата обращения: 09.04.2025). – Режим доступа: свободный.

17. PyTorch Documentation [Электронный ресурс]. – URL: https://pytorch.org/docs/stable/index.html (дата обращения: 15.04.2025). – Режим доступа: свободный.

18. Python win32 (pywin32) Documentation [Электронный ресурс]. – URL: https://pypi.org/project/pywin32/ (дата обращения: 15.05.2025). – Режим доступа: свободный.

19. REG.RU: Компьютерное зрение [Электронный ресурс]. – URL: https://www.reg.ru/blog/kompyuternoe-zrenie-chto-eto-gde-primenyaetsya/ (дата обращения: 14.05.2025). – Режим доступа: свободный.

20. TensorFlow Documentation [Электронный ресурс]. – URL: https://www.tensorflow.org/api_docs (дата обращения: 15.04.2025). – Режим доступа: свободный.

21. Tkinter Documentation [Электронный ресурс]. – URL: https://docs.python.org/3/library/tkinter.html#module-tkinter (дата обращения: 09.04.2025). – Режим доступа: свободный.

22. wxPython Documentation [Электронный ресурс]. – URL: https://wxpython.org/ (дата обращения: 09.04.2025). – Режим доступа: свободный.

23. Топ-10 лучших GUI библиотек Python 2024: обзор и сравнение [Электронный ресурс] // Яндекс.Дзен. – URL: https://dzen.ru/a/ZwJIzGBHnnxR-RVm (дата обращения: 09.04.2025). – Режим доступа: свободный.

24. Четыре способа написать Hello world, или инструменты для создания GUI на Python [Электронный ресурс] // Habr. – URL: https://habr.com/ru/companies/selectel/articles/750146/ (дата обращения: 09.04.2025). – Режим доступа: свободный.

ПРИЛОЖЕНИЕ

Ссылка на github репозиторий с кодом проекта: