DOI: 10.24412/2077-8481-2026-1-105-116

УДК 004.93

Д. В. РОГОЛЕВ, *канд. физ.-мат. наук*
А. С. ГОЛЯС
И. И. МИЩЕНКО
Белорусско-Российский университет (Могилев, Беларусь)

ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ КОМПЬЮТЕРНОГО ЗРЕНИЯ И МАШИННОГО ОБУЧЕНИЯ ДЛЯ РАСПОЗНАВАНИЯ И ПЕРЕВОДА ЖЕСТОВОГО ЯЗЫКА

Аннотация

С помощью технологий компьютерного зрения и машинного обучения на языке программирования Python разработаны приложение Gestotalk для перевода жестового языка в текст и речь, способствующее улучшению коммуникации между людьми с нарушениями слуха и их окружением, и собственная библиотека жестов. Реализована система синтеза речи для преобразования текста в речь.

Ключевые слова:

компьютерное зрение, машинное обучение, программирование, оптическое распознавание жестов.

Для цитирования:

Роголев, Д. В. Применение технологий компьютерного зрения и машинного обучения для распознавания и перевода жестового языка / Д. В. Роголев, А. С. Голяс, И. И. Мищенко // Вестник Белорусско-Российского университета. – 2026. – № 1 (90). – С. 105–116.

Введение

В современном мире эффективная коммуникация играет ключевую роль во всех сферах человеческой деятельности. Серьёзные трудности в общении возникают из-за различий в коммуникативных потребностях, особенно между теми, кто воспринимает речь на слух, и теми, кто не слышит или слышит частично. Жестовый язык является основным средством общения для людей с нарушениями слуха, но он остается недоступным для большинства слышащих людей, что создает препятствия в их взаимодействии.

По данным Всемирной организации здравоохранения (ВОЗ), более 466 млн человек во всем мире имеют инвалидность, связанную со слухом, и эта цифра продолжает расти [1], что подчеркивает важность создания технологических решений, способствующих инклюзии и улучшению качества жизни такой категории населения.

Развитие технологий компьютерного зрения и машинного обучения открывает новые возможности для преодоления этих коммуникационных барьеров. Современные алгоритмы позволяют автоматически распознавать жесты с высокой точностью, что становится основой для создания инновационных приложений.

На базе этих технологий разработано приложение Gestotalk, которое автоматически распознает жесты русского дактильного алфавита и переводит их в текст и речь. Данное решение направлено на упрощение общения между людьми с разными коммуникативными потребностями и создание более доступной среды для интеграции людей с нарушениями слуха.

Актуальность проводимого исследования определяется несколькими факторами:

– социальной необходимостью улучшения качества жизни людей с нарушениями слуха;

– развитием технологий искусственного интеллекта, создающих но-

вые возможности для решения задач автоматического распознавания жестов;

– необходимостью создания доступных и эффективных средств коммуникации для инклюзивного общества;

– возрастающим интересом к использованию технологий машинного обучения в гуманитарных областях.

Используемые технологии

Компьютерное зрение – это область искусственного интеллекта, которая позволяет компьютерам «видеть» и интерпретировать визуальную информацию. OpenCV (Open Source Computer Vision Library) – библиотека, предназначенная для обработки изображений и видео [2]. В приложении Gestotalk OpenCV используется для захвата изображений с веб-камеры и предварительной обработки кадров, что улучшает качество распознавания жестов.

Для более точного распознавания жестов в Gestotalk дополнительно применяется библиотека Mediapipe, разработанная Google [3]. Задача MediaPipe Hand Landmarker позволяет обнаруживать ориентиры рук на изображении. Эта задача оперирует данными изображения с моделью машинного обучения (Machine Learning – ML) в виде статических данных или непрерывного потока и выводит ориентиры рук в координатах изображения, ориентиры рук в мировых координатах и направление руки (левая/правая рука) нескольких обнаруженных рук. Hand Landmarker использует комплект моделей с двумя упакованными моделями: модель обнаружения ладоней и модель обнаружения ориентиров рук.

В Gestotalk Hand Landmarker используется для определения 21 ключевой точки на руке (рис. 1), что является основой для анализа жестов. Благодаря своим алгоритмам Mediapipe обеспечивает высокую точность и быстроту обработки данных, что критически важно для работы в режиме реального времени.

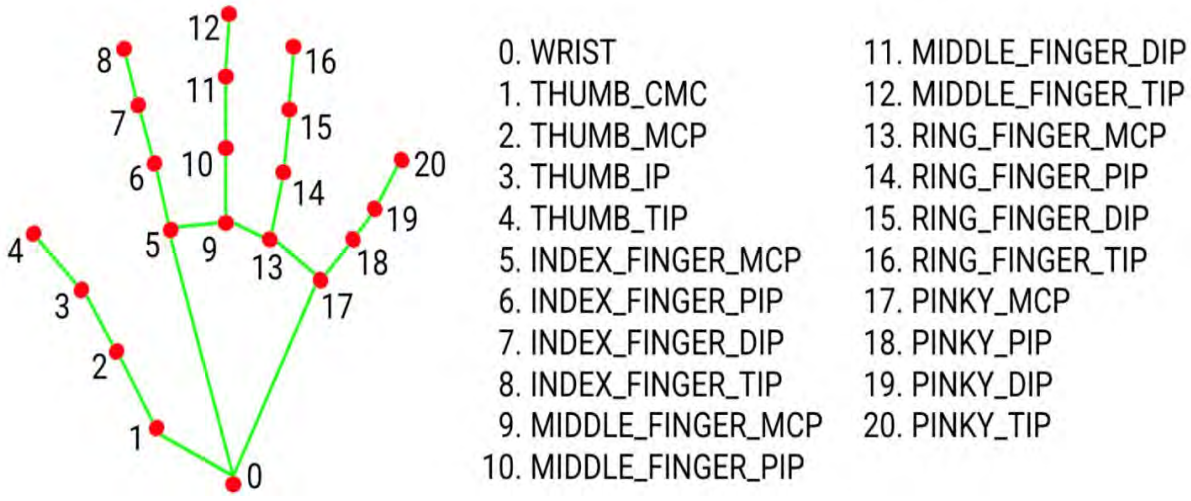

Рис. 1. Схема ключевых точек на руке в Mediapipe Hand Landmarker

Принципы работы Gestotalk

Исходный видеопоток в реальном времени захватывается с веб-камеры и первоначально обрабатывается с помощью библиотеки компьютерного зрения OpenCV. Затем подготовленный видеопоток последовательно, кадр за

кадром, передается в фреймворк Mediapipe. Эта библиотека анализирует переданные кадры и на каждом из них определяет расположение ключевых точек (рис. 2).

MediaPipe использует ‘Adaptive KeyPoint Estimation’, чтобы точно обнаруживать видимые ключевые точки на руке, а также восстанавливать скрытые точки с помощью машинного обучения, создавая более полную и точную модель объекта даже при частичном перекрытии (рис. 3).

```
import cv2
import numpy as np
import mediapipe as mp
import time
import os

cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
cap.set(10, 100)

mpHands = mp.solutions.hands
hands = mpHands.Hands(False)
npDraw = mp.solutions.drawing_utils

pTime = 0
cTime = 0

while True:
    success, img = cap.read()
    img = cv2.flip(img,1)

    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = hands.process(imgRGB)
    if results.multi_hand_landmarks:
        for handLms in results.multi_hand_landmarks:
            for id, lm in enumerate(handLms.landmark):
                h,w,c = img.shape
                cx, cy = int(lm.x*w), int(lm.y*h)
            npDraw.draw_landmarks(img, handLms, mpHands.HAND_CONNECTIONS)


    cTime = time.time()
    fps = 1/(cTime-pTime)
    pTime = cTime
    cv2.putText(img, str(int(fps)),(10,30), cv2.FONT_HERSHEY_PLAIN, 2, (255

    cv2.imshow('Mediapipe', img)
    if cv2.waitKey(20) == 27:
        break

cv2.destroyWindow()
cap.release()
cv2.waitKey(1)
```

Рис. 2. Код для запуска Hand Detection с настройками на языке программирования Python

а) *б*)

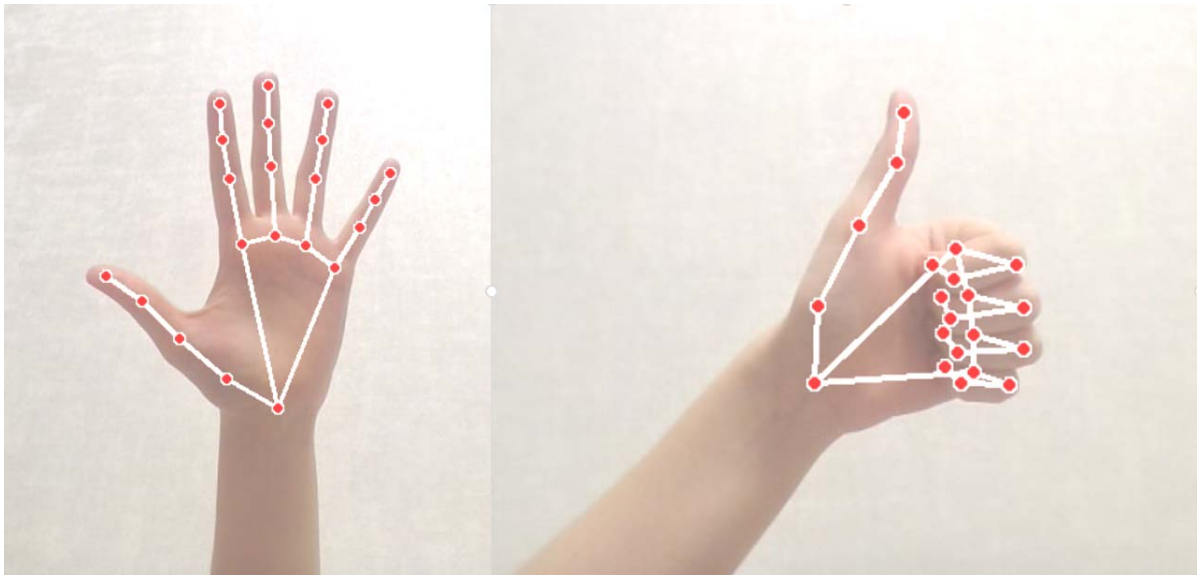

Рис. 3. Отображение маски Mediapipe: *а* – явное отображение ключевых точек; *б* – восстановленное отображение ключевых точек при частичном перекрытии

Для классификации жестов на основе данных, полученных с помощью компьютерного зрения, применяются методы сравнения жестов с заранее сохраненными образцами, которые хранятся в собственной библиотеке жестов. Для создания библиотеки жестов была разработана программа hands.py, которая анализировала положение руки в реальном времени и сохраняла координаты ключевых точек для каждого жеста. Библиотека имеет вид файла JSON, в котором каждая буква русского дактильного алфавита (кроме ё, й, щ) записана с помощью 21-го ручного ориентира, представляющих собой реальные трехмерные координаты в метрах с началом координат в геометрическом центре руки (нулевая точка) (рис. 4).

Координаты x и y нормализуются на [0, 0, 1, 0] по ширине и высоте изображения соответственно. Координата z представляет глубину ориентира, при этом глубина на запястье является началом координат. Чем меньше значение, тем ближе ориентир к камере. Величина z использует примерно тот же масштаб, что и x [3].

Для успешного распознавания и дальнейшей классификации жестов важно правильно настроить систему обработки изображений. Инициализация объекта `Hands` в MediaPipe – это настройка детектора рук для точного распознавания жестов. Она включает задание параметров модели (статической или отслеживающей), уровня сложности, порогов доверия и количества одновременно обрабатываемых рук, что напрямую влияет на качество последующей классификации жестов. Инициализация объекта ‘Hands’ в MediaPipe осуществляется следующим образом (рис. 5).

"\u0430": [[0.7687631249427795, 0.7493795156478882, -5.735094532610674e-07], [0.6965683698654175, 0.684140145778656, -0.013368280604481697], [0.6630071997642517, 0.6132872700691223, -0.03166564553976059], [0.6374495625495911, 0.5657929182052612, -0.052772920578718185], [0.6553453803062439, 0.5352916121482849, -0.06490202248096466], [0.7867124080657959, 0.46962255239486694, 0.009970247745513916], [0.6825741529464722, 0.4632883071899414, -0.029199218377470 97], [0.6852849721908569, 0.5410352945327759, -0.05853341147303581], [0.7172470688819885, 0.5557920336723328, -0.07657507061958313], [0.8010397553443909, 0.4707182049751282, 0.0004689445777330 5476], [0.6741148233413696, 0.47173523902893066, -0.041091982275247574], [0.6837798357009888, 0.5558209419250488, -0.0535750687122345], [0.7267279624938965, 0.5609430074691772, -0.055520329624414444], [0.8014572858810425, 0.47854623198509216, -0.015255311504006386], [0.6812928915023804, 0.4838281571865082, -0.04887363687157631], [0.686193585395813, 0.5637445449829102, -0.041156597435474396], [0.7276929020881653, 0.5731639266014099, -0.030017765238881 11], [0.7883394956588745, 0.49096792936325073, -0.03449718654155731], [0.6928587555885315, 0.5071230530738831, -0.056325662881135 94], [0.6948865056037903, 0.5670979022979736, -0.054181527346372604], [0.7297654151916504, 0.5822970271110535, -0.04647864401340 4846]]

Рис. 4. Запись буквы «а» в файле JSON

```
with mp.solutions.hands.Hands(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as hands:
```

Рис. 5. Инициализация объекта Hands

В этом коде задаются два ключевых параметра: `min_detection_confidence` и `min_tracking_confidence`. Первый параметр определяет минимальный уровень уверенности для обнаружения руки. Значение 0,5 означает, что алгоритм будет считать руку обнаруженной только в том случае, если уверенность составляет 50 % или больше.

Второй параметр устанавливает минимальный уровень уверенности для отслеживания руки после ее обнаружения. Значение 0,5 также указывает на необходимость уверенности в 50 % и выше. Эти настройки позволяют адаптировать алгоритм к различным условиям, улучшая его точность и надежность в распознавании жестов.

В приложении Gestotalk распознавание жестов осуществляется с помощью двух ключевых функций: `normalize_gesture` и `compare_gestures`. Эти функции работают в тандеме для обеспечения точности и надежности в классификации жестов.

Функция `normalize_gesture` (рис. 6) предназначена для нормализации координат жеста относительно за-

пястья. Это необходимо для того, чтобы система могла корректно распознавать жесты независимо от их положения на экране. Функция вычисляет относительные координаты каждой точки руки относительно координат запястья (точки 0 в списке landmark), нормализуя их по ширине и высоте изображения. В результате получается универсальное представление жеста, которое можно использовать для дальнейшего сравнения.

```
def normalize_gesture(gesture_data):
    wrist_coord = np.array(gesture_data[0][:2])
    normalized_gesture = []
    for x, y, z in gesture_data:
        normalized_x = (x - wrist_coord[0]) / (max(gesture_data,
key=lambda p: p[0])[0] - min(gesture_data, key=lambda p: p[0])[0])
        normalized_y = (y - wrist_coord[1]) / (max(gesture_data,
key=lambda p: p[1])[1] - min(gesture_data, key=lambda p: p[1])[1])
        normalized_gesture.append((normalized_x, normalized_y, z))
    return normalized_gesture
```

Рис. 6. Функция `normalize_gesture` на языке программирования Python

Функция `compare_gestures` (рис. 7) сравнивает текущий жест с заранее сохраненными образцами. Она рассчитывает расстояние между нормализованными координатами текущего жеста и сохраненными жестами. Если расстояние между текущим и сохраненным жестами меньше заданного порога, равного 30 %, функция возвращает наиболее близкое совпадение.

```
# Увеличиваем 'погрешность(предел)' (threshold=0.3) определения жеста до
30%

def compare_gestures(gesture_data, saved_gestures, threshold=0.3):
    normalized_gesture = normalize_gesture(gesture_data)
    gesture_distances = []
    for name, saved_gesture in saved_gestures.items():
        normalized_saved_gesture = normalize_gesture(saved_gesture)
        max_error = max([np.sqrt((gd[0] - sg[0])**2 + (gd[1] - sg[1])**2 +
(gd[2] - sg[2])**2) for gd, sg in zip(normalized_gesture,
normalized_saved_gesture)])
        gesture_distances.append((name, max_error))
    gesture_distances.sort(key=lambda x: x[1])
    return gesture_distances[0] if gesture_distances[0][1] < threshold
else None
```

Рис. 7. Функция `compare_gestures` на языке программирования Python

Функция `compare_gestures` использует результаты работы функции `normalize_gesture`, чтобы обеспечить точное и надежное распознавание жестов в приложении (рис. 8). Функция `normalize_gesture` принимает на вход координаты текущего жеста и нормализует их относительно запястья, что позволяет устранить влияние положения руки на экране. Нормализация

включает вычисление относительных координат каждой точки руки относительно запястья и их масштабирование по ширине и высоте изображения, в результате чего получается универсальное представление жеста, которое не зависит от его абсолютного положения и размера. Далее функция `compare_gestures` использует эти нормализованные координаты для сравнения текущего жеста с заранее сохраненными образцами. Она выполняет нормализацию как входящего жеста, так и сохраненных жестов, чтобы сравнивать их в единой системе координат. Затем рассчитываются расстояния между нормализованными координатами текущего жеста и каждого из сохраненных жестов. Полученные расстояния сортируются и выбирается наиболее близкий жест, если расстояние меньше заданного порога, например 30 %. Такой подход обеспечивает высокую точность и надежность системы распознавания жестов, делая её более устойчивой к различиям в выполнении жестов разными пользователями и позволяя эффективно справляться с большим количеством сохраненных образцов, быстро находя наиболее близкое совпадение. Таким образом, взаимодействие двух функций является ключевым элементом системы Gestotalk, давая возможность ей корректно интерпретировать движения рук пользователей в различных условиях.

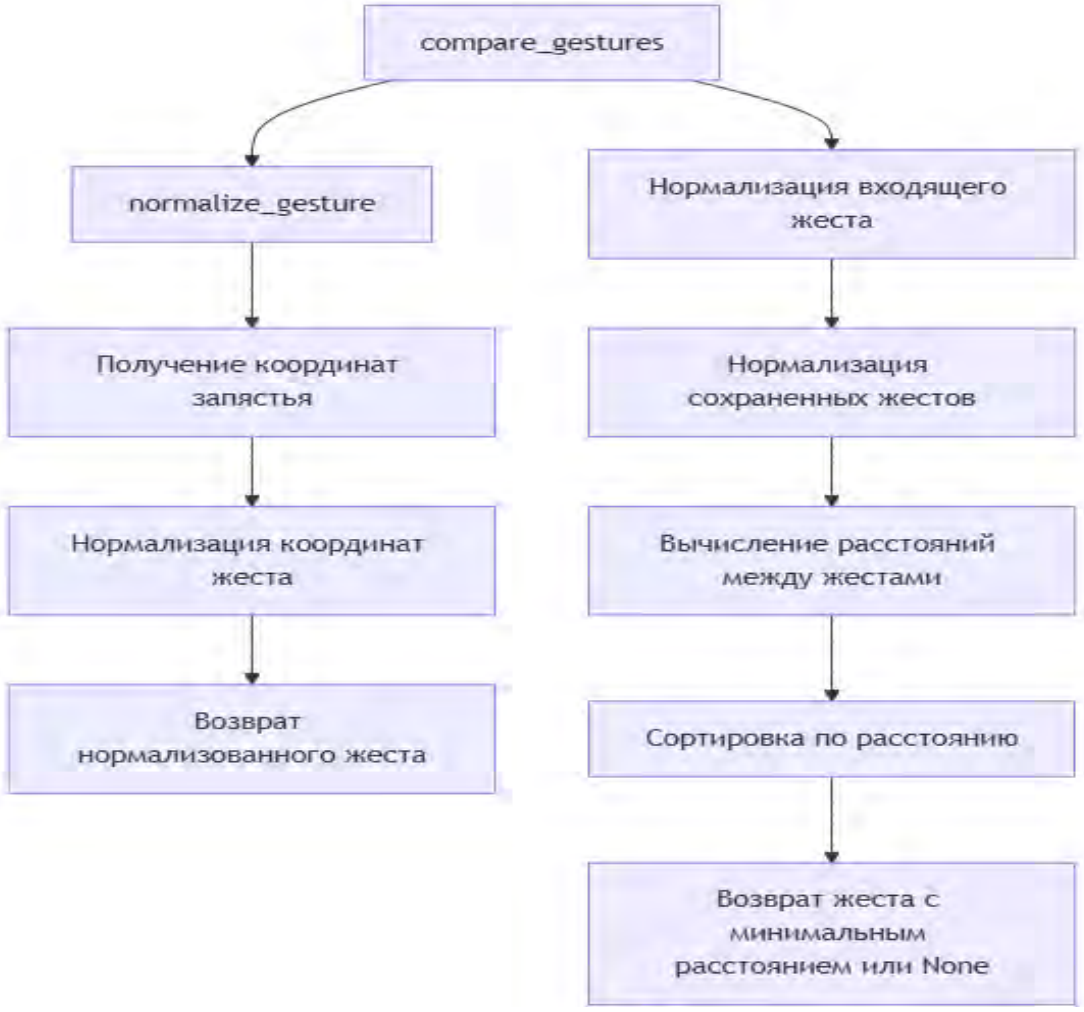

Рис. 8. Схема работы функций `normalize_gesture` и `compare_gestures`

Для оптимизации процесса распознавания жестов внедрена многопоточность, используя соответствующую библиотеку threading в Python. Библиотека ‘threading’ позволяет создавать и управлять потоками, что особенно полезно в задачах, связанных с обработкой большого количества данных в реальном времени [4]. В приложении реализовано несколько потоков, каждый из которых отвечает за распознавание жестов на обеих руках. Это значительно снизило задержку и повысило общую производительность. Применение библиотеки также привело к ускорению обработки данных и улучшению как точности распознавания, так и корректности получаемых результатов.

Многопоточность внедрена в первую очередь для одновременного распознавания жестов на обеих руках и для назначения каждой руке определённых функции. Система распознает жесты правой руки, соответствующие буквам русского алфавита, в то время как левая рука используется для выполнения функций обработки распознанных жестов (букв) правой руки (рис. 9). Функции левой руки, такие как ввод, пробел и удаление, применяются для внесения и обработки текста в текстовом поле, которое является частью графического интерфейса (GUI), разработанного с использованием PyQt5.

Рис. 9. Дактильный алфавит (правая рука) и жесты – функции для работы с распознанными жестами правой руки (левая рука)

PyQt5 – это набор привязок Python для библиотеки Qt, который позволяет создавать кроссплатформенные приложения с графическим интерфейсом [5]. Всё GUI для приложения Gestotalk реализовано на PyQt5, что обеспечивает

удобный и интуитивно понятный интерфейс для пользователей. Интерфейс включает в себя текстовое поле для отображения распознанных букв, кнопки для сохранения текста в формате .txt, записи звука в формате MP3 и озвучивания текста из текстового поля, а также инструкцию по применению приложения (рис. 10).

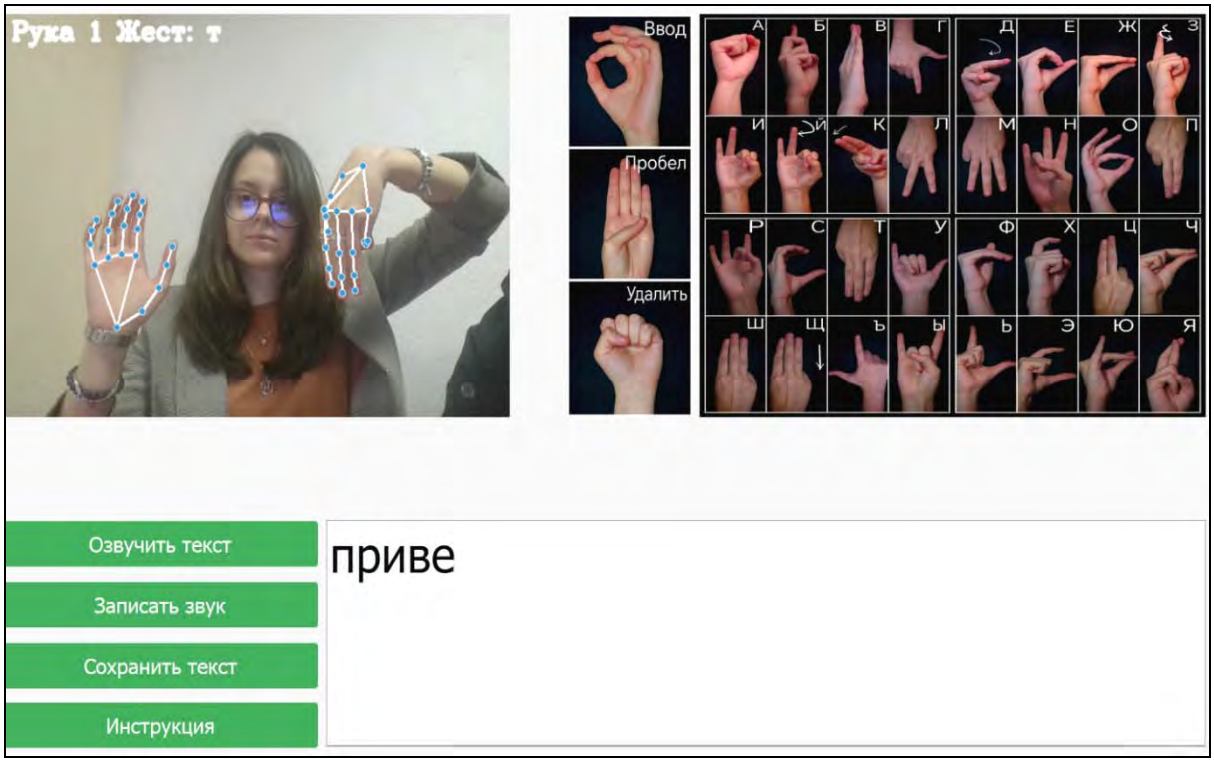

Рис. 10. Скриншот приложения Gestotalk

Для озвучивания текста из текстового поля и дальнейшей записи использовалась библиотека для синтеза речи из текста gTTS (Google Text-to-Speech) [6].

Многопоточность в контексте GUI позволила избежать задержек в работе интерфейса при выполнении длительных операций, таких как распознавание жестов или озвучивание текста. Это обеспечило более плавное и отзывчивое взаимодействие пользователя с приложением.

Приложение Gestotalk работает следующим образом (рис. 11). С помощью OpenCV оно захватывает видеопоток с веб-камеры, что позволяет пользователю вводить жесты в реальном времени. Mediapipe Hand Landmarker обрабатывает каждый кадр и определяет ключевые точки на руке. Эти данные передаются в функцию сравнения жестов. С помощью методов машинного обучения приложение сравнивает полученные данные с сохраненными жестами из библиотеки JSON. Признается жест, который наиболее близок к текущему. Результаты распознавания отображаются на экране. При обнаружении жеста «ввод» левой руки распознанный в тот же момент жест правой руки записывается в текстовое поле. При необходимости текст может быть озвучен, записан с помощью gTTS и сохранён в форматах .txt и MP3.

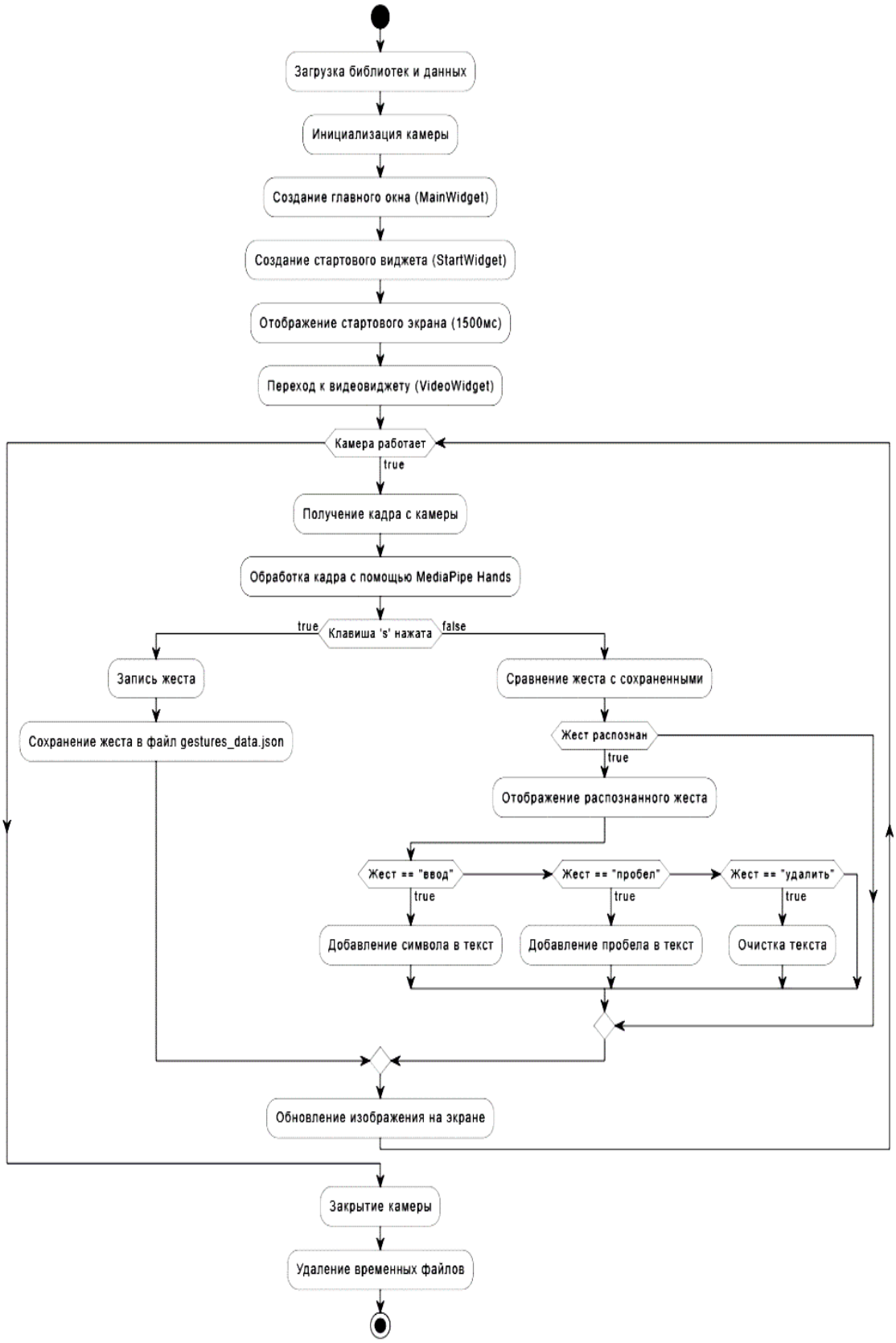

Рис. 11. Блок-схема принципа работы приложения Gestotalk

Заключение

Разработанное приложение Gestotalk демонстрирует эффективное применение технологий компьютерного зрения и машинного обучения для распознавания жестового языка. Оно может способствовать улучшению коммуникации между людьми с нарушениями слуха и их окружением, а также предоставляет гибкие возможности для адаптации под конкретные потребности пользователей. Помимо этого, приложение может активно применяться в волонтерской деятельности, особенно в сфере обучения жестовому языку, что помогает создавать более доступную среду для инклюзии людей с нарушениями слуха. Реализация собственной библиотеки жестов и использование многопоточности значительно повышают его функциональность и производительность.

На данный момент проект уже достиг стадии минимально жизнеспособного продукта (MVP) и его функциональный прототип успешно прошел проверку в реальных условиях. Проект активно участвовал в таких мероприятиях, как областной турнир по программированию «Командный хакатон CodingFest-9» (г. Могилев), где он получил признание, завоевав диплом II степени, конкурс стартапов Белорусско-Российского университета, форум инновационного предпринимательства Startup Generation (г. Минск), форум Digital Expo TIBO – 2025 (г. Минск), а также выставка-конкурс «100 инноваций молодых ученых» в рамках Фестиваля науки (г. Минск). Проект стал победителем молодежного проекта «100 идей для Беларуси» в категории «Общество и социальная сфера» в студенческой категории участников. Это подтверждает высокую эффективность и перспективность разработки.

В будущем планируется расширение функциональности, включая поддержку большего количества жестов и языков, а также улучшение точности распознавания.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. World report on hearing / World Health Organization. – Geneva, 2021.
2. OpenCV – Open Computer Vision Library. – 2025. – URL: https://opencv.org (date of access: 10.03.2025).
3. MediaPipe solutions guide // Google AI for Developers. – 2025. – URL: https://developers.google.com/mediapipe (date of access: 11.03.2025).
4. Threading – Thread-based parallelism // Python 3.13.2 documentation. – 2025. – URL: https://docs.python.org/3.13/library/threading.html (date of access: 12.03.2025).
5. Руководство по PyQt5 / Pythonist. – 2025. – URL: https://pythonist.ru/rukovodstvo-po-pyqt5/amp/ (дата обращения: 13.03.2025).
6. **Durette, P. N.** gTTS (Google Text-to-Speech) / P. N. Durette. – 2025. – URL: https://pypi.org/project/gTTS/ (date of access: 14.03.2025).

Статья сдана в редакцию 16 октября 2025 года

Контакты:
d-rogolev@tut.by (Роголев Дмитрий Владимирович);
y367152@gmail.com (Голяс Анастасия Сергеевна);
sombra74@yandex.ru (Мищенко Илья Игоревич).

D. V. ROGOLEV*, *A. S. GOLYAS*, *I. I. MISHCHENKO

APPLICATION OF COMPUTER VISION AND MACHINE LEARNING TECHNOLOGIES FOR SIGN LANGUAGE RECOGNITION AND TRANSLATION

Abstract

Using computer vision and machine learning technologies, the GestoTalk application for translating sign language into text and speech, along with a custom gesture library, was developed in the Python programming language. The application facilitates communication between people with hearing impairments and those around them. A speech synthesis system was implemented for text-to-speech conversion.

Keywords:

computer vision, machine learning, programming, optical gesture recognition.

For citation:

Rogolev, D. V. Application of computer vision and machine learning technologies for sign language recognition and translation / D. V. Rogolev, A. S. Golyas, I. I. Mishchenko // Belarusian-Russian University Bulletin. – 2026. – № 1 (90). – P. 105–116.