УДК 004.8

ПРИМЕНЕНИЕ ТЕХНОЛОГИИ MEDIAPIPE В СИСТЕМАХ АВТОМАТИЗИРОВАННОГО ПРОКТОРИНГА

Чернопенев М.С., студент гр. ИИм-231, VI курс
Кононцов А.А.., студент гр. ИИм-231, VI курс
Ванеев О.Н. доцент (к.н.) кафедры ИиАПС
Кузбасский государственный технический университет имени Т. Ф. Горбачева г. Кемерово

При создании системы, предполагающей работу с машинным зрением, неизбежно встает вопрос выбора подходящей технологии. В настоящее время существует множество технологий, которые позволяют решать задачи машинного зрения и уже зарекомендовали себя в различных сферах. Например, OpenCV активно используется в разработке систем компьютерного зрения для обработки изображений и видео, YOLO (You Only Look Once) применяется для детекции объектов в реальном времени, а TensorFlow и PyTorch — для создания и обучения моделей глубокого обучения, используемых в сложных проектах. Эти технологии находят применение в медицине (анализ медицинских изображений), промышленности (автоматизация процессов и контроль качества), безопасности (системы видеонаблюдения) и даже в развлекательной индустрии (анимация и дополненная реальность).

В рамках разработки системы прокторинга, одной из ключевых задач которой является мониторинг действий пользователя с использованием технологий машинного зрения, важным аспектом становится выбор универсального и гибкого инструмента.

Целью данной работы является описание технологии MediaPipe и обоснование ее преимуществ как подходящего решения для разработки и дальнейшего функционирования системы прокторинга.

Системы прокторинга представляют собой технологические решения, предназначенные для удаленного контроля за процессом проведения экзаменов, тестирований или других мероприятий, где требуется наблюдение за пользователями. Эти системы активно используются в образовательных учреждениях, корпоративном обучении и сертификационных центрах, обеспечивая соблюдение правил и предотвращение нарушений [1].

Ключевыми функциями систем прокторинга являются мониторинг активности пользователя через веб-камеру и микрофон, запись экрана,

обнаружение подозрительных действий (например, открытие запрещенных приложений или вкладок). Для эффективного выполнения этих задач необходимо фиксировать и анализировать поведение пользователя, включая его движения, мимику и наличие посторонних объектов или лиц в кадре.

Важнейшим компонентом таких систем является применение технологий машинного зрения. С их помощью осуществляется отслеживание движений глаз и головы для выявления попыток подсказок или использования запрещенных материалов, а также детекция посторонних объектов или лиц в зоне наблюдения. Машинное зрение позволяет автоматически анализировать позу и поведение пользователя, фиксируя любые аномалии, которые могут указывать на нарушение правил. Для разработки системы прокторинга эффективным выбором является использование технологии MediaPipe.

MediaPipe — это технология, разработанная для быстрой интеграции решений на основе машинного обучения (ML). Она предоставляет разработчикам высокоуровневые инструменты для обработки мультимедийных данных, таких как изображения, видео и аудио, и позволяет создавать сложные приложения, не начиная с нуля. MediaPipe опирается на гибкую модульную архитектуру, что делает ее подходящей для создания графов обработки данных. В основе MediaPipe лежит принцип конвейера (pipeline), который позволяет обрабатывать данные на лету.

MediaPipe — это мощная платформа с открытым исходным кодом, разработанная для создания решений в области компьютерного зрения. Она предлагает широкий набор библиотек и инструментов, которые позволяют разработчикам быстро внедрять методы искусственного интеллекта и машинного обучения. Среди главных преимуществ MediaPipe — возможность работы на различных устройствах и платформах, гибкость в настройке, а также поддержка решений для задач, связанных с распознаванием жестов, лиц, движений и многого другого [2].

Для более подробного описания о преимуществах и недостатках MediaPipe сравним с одним из популярных его аналогов технология OpenPose [3].

Таблица 1

Критерий	**MediaPipe**	**OpenPose**
Производительность	Работает в реальном времени даже на устройствах с низкой мощностью (мобильные устройства).	Требует мощного оборудования, особенно GPU, что делает его недоступным для использования на смартфонах.
Простота интеграции	Легко интегрируется в проекты благодаря	Требует значительных усилий для настройки и интеграции,

	готовым графам и кросс-платформенной поддержке.	особенно для приложений на мобильных платформах.
Оптимизация ресурсов	Оптимизирован для работы с аппаратным ускорением (GPU, Edge TPU).	Высокое потребление вычислительных ресурсов даже при низком разрешении входных данных.
Кроссплатформенность	Поддерживает Android, iOS, Windows, macOS и браузеры (через WebAssembly).	Поддержка ограничивается серверными платформами (Windows, Linux) с высокой производительностью.
Простота использования	Готовые предобученные решения позволяют быстро начать разработку.	Нужны глубокие знания компьютерного зрения для настройки даже базовых функций.
Точность	Хорошая точность для задач реального времени, но может уступать в сложных условиях (например, множество людей).	Высокая точность при обработке сложных данных, но за счет производительности и ресурсов.
Документация и поддержка	Активно поддерживается Google, имеет хорошую документацию и активное сообщество разработчиков.	Документация менее дружелюбна, часто требуется разбираться в исходном коде для решения проблем.
Гибкость	Удобен для стандартных сценариев, но ограничен в кастомизации.	Предлагает больше возможностей для тонкой настройки, но требует времени и опыта.

Из таблицы 1 можно сделать вывод, что в то время, как OpenPose требует мощных серверов или игровых графических процессоров.

Рассмотрим работу двух популярных моделей, предоставляемых этой платформой: модели нахождения рук и модели нахождения точек лица. Эти решения основаны на алгоритмах машинного обучения и обеспечивают высокую точность при обработке видеопотока в реальном времени.

Модель нахождения рук в MediaPipe позволяет детектировать и отслеживать положение рук пользователя в реальном времени. Она использует нейронные сети для распознавания ключевых точек на руках, что позволяет точно определять положение и движение рук.

Принцип работы заключается в том, что модель анализирует изображения, поступающие с камеры, и использует алгоритмы глубокого обучения для нахождения ключевых точек на каждой руке. Эти точки включают 21 координату, которые определяют положение каждого пальца и запястья.

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

# Подключаем камеру
cap = cv2.VideoCapture(0)
cap.set(3, 640) # Width
cap.set(4, 480) # Lenght
cap.set(10, 100) # Brightness

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

pTime = 0
cTime = 0

    #Зацикливаем получение кадров от камеры
while True:
    success, img = cap.read()
    img = cv2.flip(img,1) # Mirror flip

    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)
               # print(id, lm)
                if  id == 8 or id == 12:
                    cv2.circle(img, (cx,cy),10,(255,0,255),cv2.FILLED)

            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, (255,0,0), 2) # ФреймРейт

    cv2.imshow('python', img)
    if cv2.waitKey(20) == 27: # exit on ESC
        break

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

Рисунок 1 – Отслеживание движение рук.

На рисунке 1 изображен код, в котором используются библиотеки OpenCV, NumPy и MediaPipe для захвата изображения с камеры, обработки рук с помощью модели из MediaPipe и отображения результатов в реальном

времени. Он открывает камеру с разрешением 640x480 пикселей, настраивает яркость камеры на максимальное значение, и затем захватывает каждый кадр. С помощью модели из MediaPipe код обрабатывает изображение, находя руки и отображая ключевые точки (landmarks) на них, выделяя определенные точки, такие как кончики пальцев (идентификаторы 8 и 12). Для улучшения визуализации, на этих точках рисуются круги.

Вся суть заключается в поиске 3 значений в пространстве для каждый из 21 точки руки (рисунок 2-3).

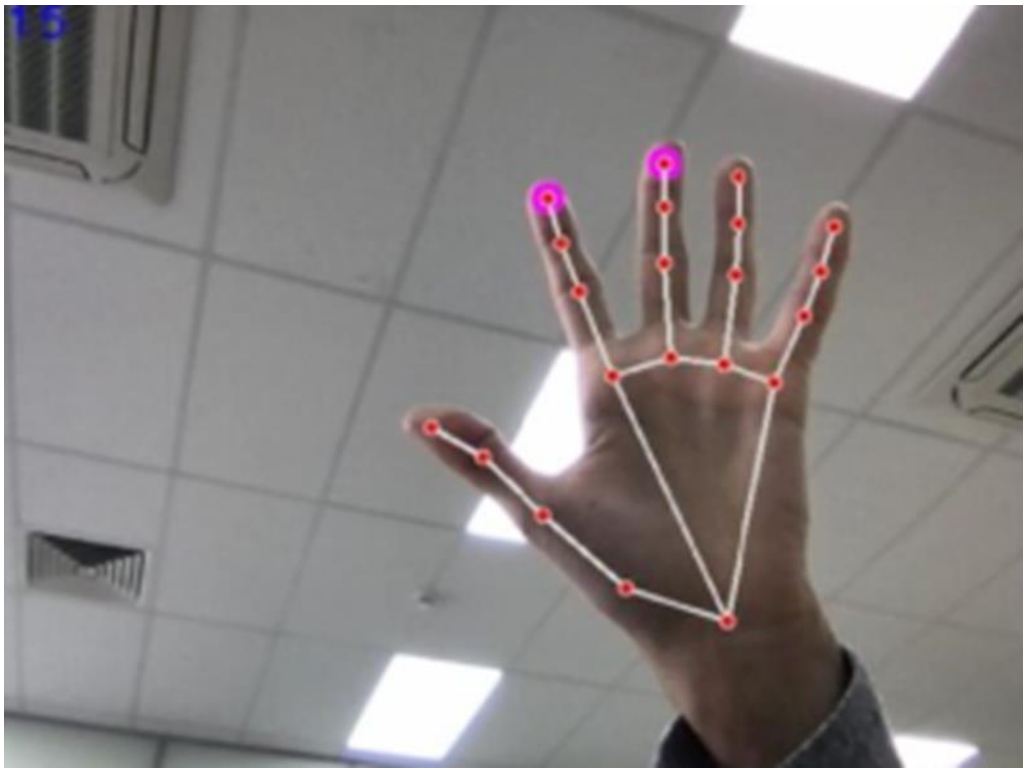

Рисунок 2 – Поиск 3 значений в пространстве для каждый из 21 точки руки.

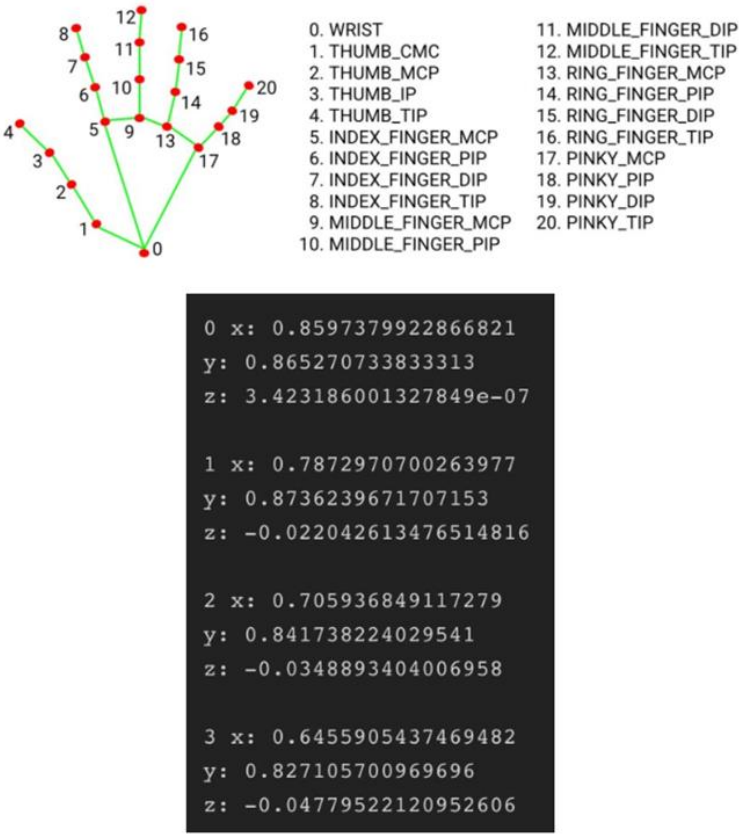

Рисунок 3 – Поиск 3 значений в пространстве для каждый из 21 точки руки.

Зная точное расположение этих точек, можно строить различные модели. Детектирование рук в MediaPipe можно настроить с высокой гибкостью, варьируя точность определения, а также устанавливая ограничения на количество рук, которые могут быть обнаружены в кадре [4].

Модель нахождения точек лица в MediaPipe, известная как Face Mesh, позволяет определять 468 ключевых точек на лице пользователя. Это решение основано на анализе изображения с камеры и использует нейронные сети для нахождения точек, которые определяют форму и выражения лица.

Принцип работы заключается в том, что модель обрабатывает входное изображение, используя алгоритмы детекции и глубинные нейронные сети для нахождения ключевых точек на лице. Эти точки включают контуры глаз, бровей, носа, рта и челюсти, что позволяет получать подробное представление о положении лица.

```python
import cv2
import mediapipe as mp
import time

# Подключаем камеру
cap = cv2.VideoCapture(0)
cap.set(3, 640)  # Ширина
cap.set(4, 480)  # Высота
cap.set(10, 100)  # Яркость

mp_drawing = mp.solutions.drawing_utils
mp_holistic = mp.solutions.holistic

pTime = 0

# Зацикливаем получение кадров от камеры
while True:
    with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
        ret, frame = cap.read()
        if not ret:
            break

        # Перекодируем изображение в RGB
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        # Выполняем распознавание
        results = holistic.process(image)
        # Перекодируем изображение обратно в BGR
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # Рисуем ориентиры лица
        if results.face_landmarks:
            mp_drawing.draw_landmarks(
                image,
                results.face_landmarks,
                mp_holistic.FACEMESH_TESSELATION,
                mp_drawing.DrawingSpec(color=(80, 110, 10), thickness=1, circle_radius=1),
                mp_drawing.DrawingSpec(color=(80, 256, 121), thickness=1, circle_radius=1)
            )


        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime
        cv2.putText(image, f'FPS: {int(fps)}', (10, 30), cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2)

        cv2.imshow('Face Landmarks', image)

    if cv2.waitKey(1) == 27:  # Выход по ESC
        break

# Освобождаем ресурсы
cap.release()
cv2.destroyAllWindows()
```

Рисунок 4 – Отслеживание движение лица.

На рисунке 4 изображен код, в котором используются библиотеки MediaPipe и OpenCV для отслеживания движений лица, тела и рук в реальном времени с помощью веб-камеры. Сначала подключается камера, настраиваются параметры видео (разрешение, яркость), затем с помощью MediaPipe Holistic производится обработка каждого кадра. Кадры преобразуются в формат RGB, передаются в модель MediaPipe, которая возвращает ориентиры лица, рук и тела. Эти ориентиры визуализируются на кадре с помощью функций рисования из MediaPipe. На изображение накладывается текущий FPS (кадры в секунду), а результат выводится в окно OpenCV. Программа работает в цикле, пока пользователь не нажмет клавишу ESC, после чего закрывает окно и освобождает ресурсы камеры.

Вся суть заключается в поиске 3 значений в пространстве для каждой из 468 точки на лице (рисунок 5).

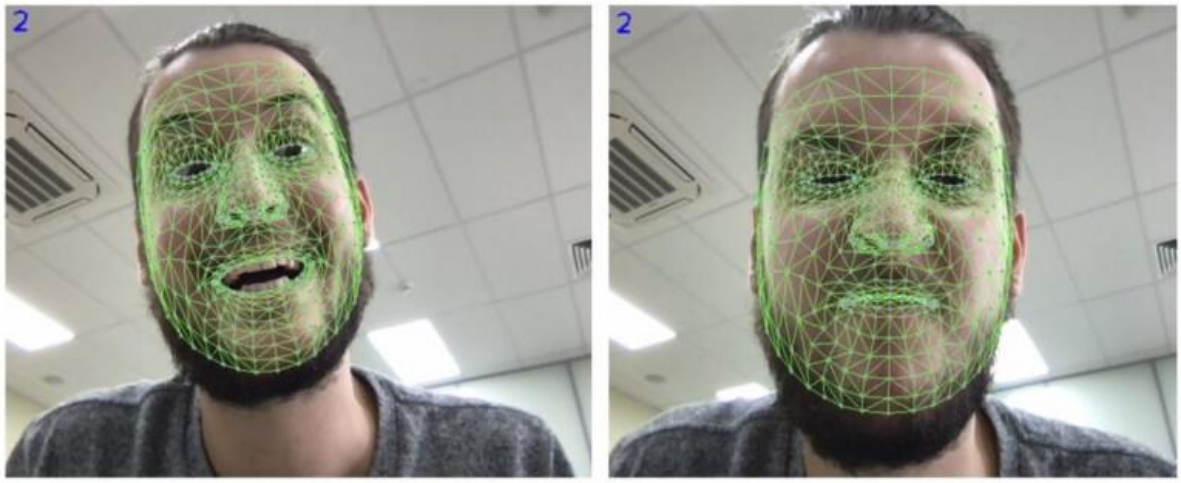

Рисунок 5 – Поиск 3 значений в пространстве для каждой из 468 точки на лице.

Именно с подобной технологией делаются все маски для лиц в социальных сетях.

MediaPipe в сочетании с другими инструментами машинного зрения представляет собой надежную основу для разработки систем прокторинга, способных решать задачи мониторинга и предотвращения нарушений при сдаче экзаменов.

Список литературы:

1. Целик М. С., Иванилов Т. А. ОСОБЕННОСТИ ПРОКТОРИНГА В СОВРЕМЕННЫХ ОБРАЗОВАТЕЛЬНЫХ ПРОЦЕССАХ // Россия: тенденции и перспективы развития. 2023. №18-2.URL:https://cyberleninka.ru/article/n/osobennosti-proktoringa-v-sovremennyh-obrazovatelnyh-protsessah
2. MediaPipe Solutions Guide [Электронный ресурс]. – Режим доступа: https://ai.google.dev/edge/mediapipe/solutions/guide?hl=ru
3. OnePose++ [Электронный ресурс]. – Режим доступа: https://github.com/zju3dv/OnePose_Plus_Plus
4. MediaPipe: библиотека для создания приложений компьютерного зрения [Электронный ресурс] // Habr. – 2021. – Режим доступа: https://habr.com/ru/articles/596043/