ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт машиностроения, материалов и транспорта

Высшая школа автоматизации и робототехники

Курсовой проект

по дисциплине «Объектно-ориентированное программирование»

«Автонаведение на голову в видеопотоке средствами компьютерного зрения с использованием нейронных сетей»

Выполнил студент

гр. 3331506/20401 _____ Варга В.В.

(подпись)

Работу принял

_____ Ананьевский М.С.

(подпись)

Санкт-Петербург

2025 г.

Оглавление

1. Теоретические сведения

Определение:
YOLO (You Only Look Once) — это сверточная нейронная сеть (CNN), предназначенная для детекции объектов в реальном времени. Отличается высокой скоростью работы за счёт однопроходной обработки изображения.

Принцип действия:

1. *Детекция объектов за один проход:*
    - Изображение делится на сетку (например, 19×19 ячеек).
    - Каждая ячейка предсказывает координаты bounding box, вероятность наличия объекта и его класс.
2. *Использование anchor boxes:*
    - Для повышения точности YOLO использует "якорные" прямоугольники — шаблоны типичных размеров объектов (например, для головы человека).
3. *Фильтрация результатов:*
    - Применяется алгоритм NMS (Non-Maximum Suppression) для удаления дублирующих предсказаний.

Определение:
MediaPipe Face Mesh — это легковесная нейронная сеть, разработанная Google для поиска ключевых точек лица в режиме реального времени. Оптимизирована для работы на мобильных устройствах и ПК.

Принцип действия:

1. *Двухэтапная обработка:*
    - Этап 1: Детекция лица с помощью модели BlazeFace. Она быстро находит bounding box лица.
    - Этап 2: Регрессия 468 ключевых точек. Сеть анализирует область лица и предсказывает 3D-координаты точек (глаза, нос, брови, лоб).
2. *Топология точек:*
    - Каждая точка имеет фиксированный номер (например, точка 10 — между бровями, 151 и 337 — границы лба).
3. *Адаптация к поворотам:*
    - Модель учитывает 3D-геометрию лица, что позволяет определять точки даже при частичном повороте головы.

2. Основная часть

Изначально идея курсовой работы появилась из желания создать программное обеспечение позволяющее наводиться на голову противника. Это ПО на языке Python будет бы основой для создания автономной турели. Главная задача ПО – поредение головы при любых условиях: Лицо видно, лица не видно, лицо скрыто и т.д. Язык выбран в виду особой удобности при работе с нейронными сетями и компьютерным зрением, а также большим набором библиотек для этих целей.

Для реализации поставленной задачи было принято решение использовать одновременно две нейронных сети, про которые ранее было упомянуто в кратких теоретических сведениях: YOLO и MediaPipe. Почему нельзя было обойтись только YOLO или вообще обойтись без использования нейронных сетей? Начнем с того, что классические средства компьютерного действительного способны на выполнение поставленной задачи – обнаружение головы в кадре, но при этом решение крайне уязвимо к смене яркости, наличию бликов и других «шумовых» факторов. Использование двух нейронных сетей обосновано тем, что YOLO обучалась на датасете людей во весь рост с выделением частей: тело, голова. За счет этого не имеет значения, надел человек маску или находится спиной к камере, найти его голову всегда возможно, но при этом за счет такого подхода падает точность распознавания и нейронную сеть можно обмануть, например курткой с кепкой, но, чтобы это поправить нужно просто добавить в код (*смотреть приложение 1*) условие на соблюдение пропорций тела, а также повысить порог распознавания за счет увеличения точности. MediaPipe же в свою очередь отвечала, за нахождение головы с более высокой точностью если лицо человека было видно. Так как нейронная сеть заточена конкретно на расписывание лиц, а также имеет возможность построения 3д модели лица, то точность действительно оказалась на высоте. Помимо того, что эта нейронная сеть очень удачно дополняет YOLO, в перспективе создания автономной турели её можно использовать как базу для лиц, по которым огонь запрещён – по союзникам.

Далее расмотрим принцип работы нейронных сетей в самом коде:

1. Загрузка предобученных моделей

YOLO("yolov8n.pt"):

Автоматическая загрузка файла модели yolov8n.pt

2. Инициализация архитектуры нейросети:

244 слоя сверточных нейронов

3.2 миллиона обучаемых параметров

Размер модели: ~6 МБ

Все тоже самое для MediaPipe, только размер легковесной модели 2.5 МБ.

Внутри этих .pt файлов находятся:

+Веса нейронной сети (числовые коэффициенты)

+Описание архитектуры слоев

+Метаданные о классах объектов

Принцип работы самих сетей, был рассмотрен в краткой теории. Возникающие неточности и погрешности, возжигаемые при работе с нейронными сетями обусловлены их малыми размерами моделей, однако, даже не смотря на это искусственный интеллект дополняющий компьютерное зрение работает превосходно, в этом можно убедиться прочитав раздел результаты работы программы, а также осознав, что хорошая работа обусловлена:

+1.2 миллионом изображений (для YOLO)

+50,000 размеченных 3D-сканов лиц (для MediaPipe)

+Более 100+ GPU-лет тренировочного времени

3. Результаты работы программы

Программное обеспечение подробно расписано с комментариями в приложении 1, но если написать планировку программы, то выглядеть она будет так:

1) Подключаем предобученные модули нейронных сетей
2) Получаем изображение с камеры
3) Преднастраиваем точность и количество лиц на распознание
4) Пытаемся найти лица с помощью MediaPipe, иначе пробуем YOLO
5) Выделаем контуры, наводимся на лоб и выводим результат поверх текущего видеопотока.

На следующей странице представлены рисунки 1,2 и 3 на которых приведены результаты различного тестирования программного обеспечения в различных случаях:

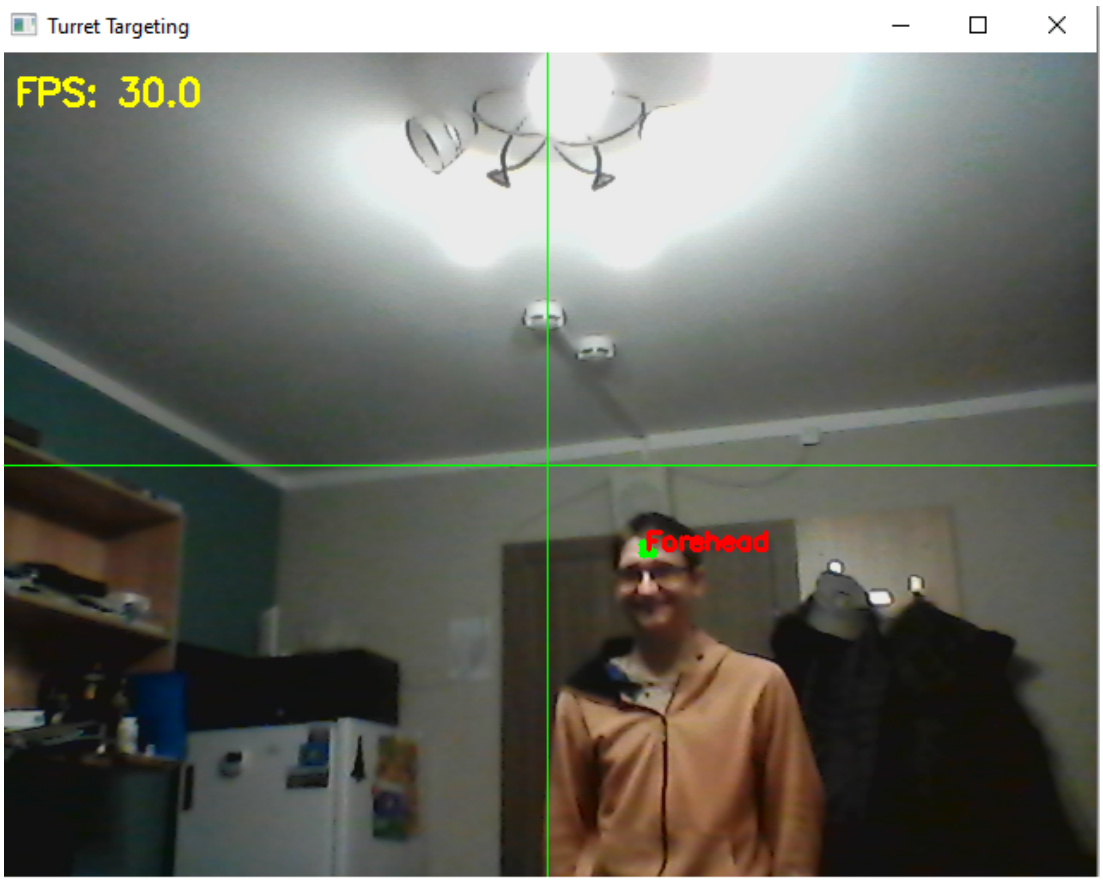

Рисунок 1 – Работа ПО при видимости лица.

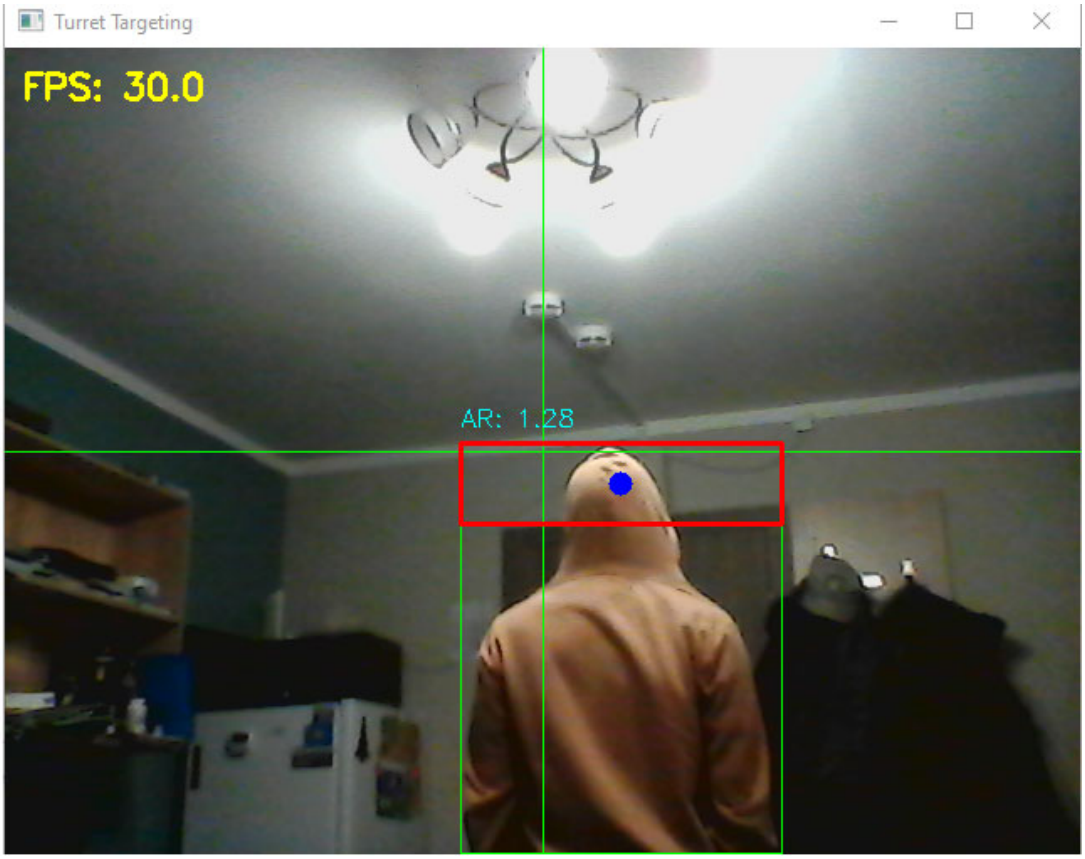

Рисунок 1 – Работа ПО при отсутствии лица.

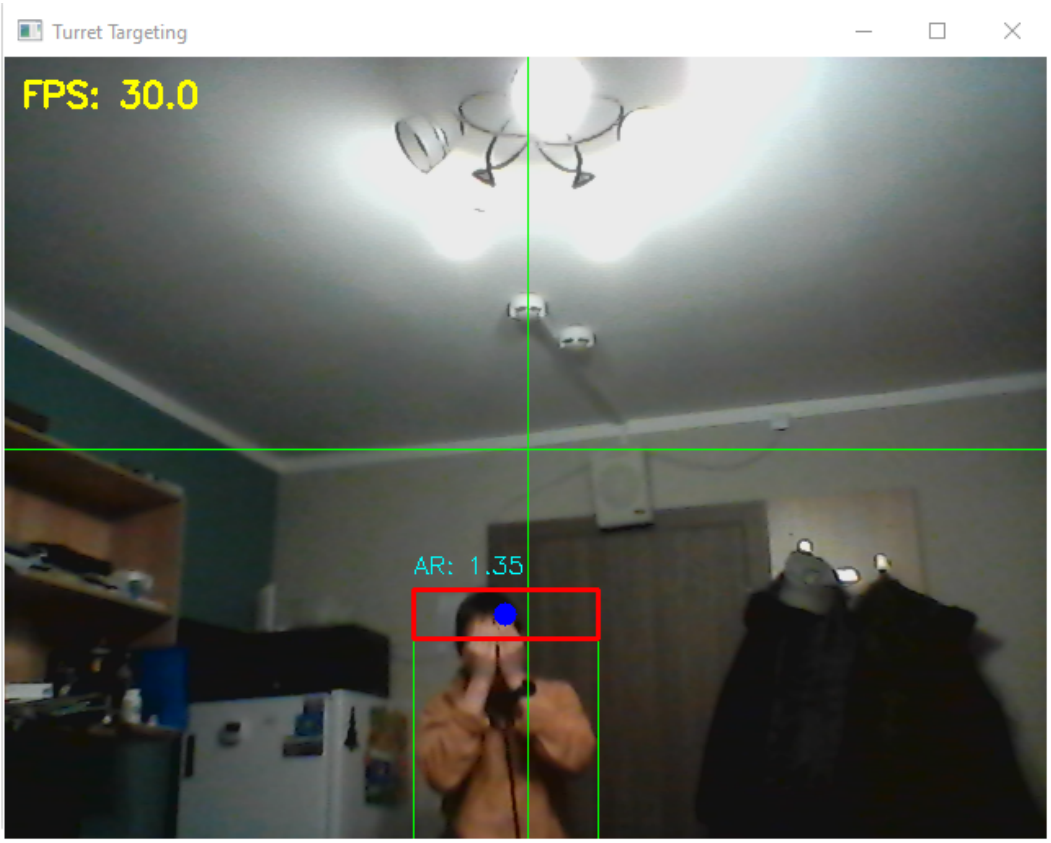

Рисунок 3 – Работа ПО при отсутствии лица на дальнем расстоянии.

4. Заключение

В процессе выполнения курсовой работы появилось базовое понимание работы нейронных сетей и их обучение, на примере MNIST модели, удалось написать ПО для нахождения головы человека в кадре при этом независимо от видимости лица, также удалось увеличить точность распознавания за счет игнорирования куртки, как видно из кадров при тестировки куртка была проигнорирована. Как итог программное обеспечение можно считать успешным.

5. Список источников

-Определение нейронной сети:
https://ru.wikipedia.org/wiki/Нейронная_сеть

-Детекция лиц и применение классификации в разделении изображения на сетку по алгоритму YOLO:
https://teta-arm.ru/detektsiya-lits-i-primenenie-klassifikatsii-v-razdelenii-izobrazheniya-na-setku-po-algoritmu-yolo/

- Распознавание маски на лице с помощью YOLO:
https://habr.com/ru/companies/skillfactory/articles/552400/

-Руководство по обнаружению ориентиров лица:
https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker?hl=ru

-Построение примитивной нейронной сети по распознаванию объектов:
https://www.youtube.com/watch?v=8mkh4uGxNfo

Приложение 1

```
import cv2
import numpy as np
from ultralytics import YOLO
import mediapipe as mp

# Инициализация моделей
model_yolo = YOLO("yolov8n.pt")
mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(static_image_mode=False, max_num_faces=3)

# Подключение камеры
cap = cv2.VideoCapture(0)

# Получаем размеры кадра только после успешного подключения
if cap.isOpened():
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))  # Современный синтаксис
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
else:
    print("Ошибка: не удалось подключиться к камере")
    exit()

while True:
    ret, frame = cap.read()
    if not ret:
        break
    processed_frame = frame.copy()

    # Проверяем размеры кадра (на случай изменения разрешения)
    current_height, current_width = frame.shape[:2]
```

```
if current_width != width or current_height != height:
    width, height = current_width, current_height

# Рисуем центральные линии (с приведением к целым числам)
center_x = width // 2
center_y = height // 2

# Вертикальная линия
cv2.line(processed_frame,
         (center_x, 0),
         (center_x, height),
         (0, 255, 0),
         thickness=1)

# Горизонтальная линия
cv2.line(processed_frame,
         (0, center_y),
         (width, center_y),
         (0, 255, 0),
         thickness=1)

results = model_yolo.predict(
    frame,
    classes=[0],          # Только люди
    conf=0.65,            # Минимальная уверенность (отсеяли куртку)
    iou=0.5,              # Подавление дублирующих боксов
    verbose=False)

for box in results[0].boxes:
    x1, y1, x2, y2 = map(int, box.xyxy[0])
    head_roi = frame[y1:y2, x1:x2]
```

```
        # Поиск ключевых точек лица через MediaPipe
        rgb_roi = cv2.cvtColor(head_roi, cv2.COLOR_BGR2RGB)
        results_mp = face_mesh.process(rgb_roi)
        if results_mp.multi_face_landmarks:
            for landmarks in results_mp.multi_face_landmarks:
                forehead_points = [landmarks.landmark[i] for i in [10, 151, 337]
                # Конвертация координат в пиксели
                h, w = head_roi.shape[:2]
                forehead_px = [(int(lm.x * w) + x1, int(lm.y * h) + y1) for lm in forehead_points]
                # Отрисовка
                for x, y in forehead_px:
                    cv2.circle(processed_frame, (x, y), 3, (0, 255, 0), -1)
                # Центр лба (примерно)
                avg_x = sum(x for x, y in forehead_px) // len(forehead_px)
                avg_y = sum(y for x, y in forehead_px) // len(forehead_px)
                cv2.putText(processed_frame, "Forehead", (avg_x, avg_y),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
        else:
            width = x2 - x1
            height = y2 – y1
            # Фильтр 1: Пропорции головы (ширина/высота в реалистичных пределах)
            aspect_ratio = height / width
            if not (1.0 < aspect_ratio < 3.0):  # Эмпирические значения
                continue  # Пропускаем нетипичные для головы объекты


            # Фильтр 2: Минимальный размер (чтобы игнорировать далекие/маленькие объекты)
            if width < 50 or height < 50:
                continue


            # Лоб: верхние 20% от высоты головы
            forehead_y1 = y1
            forehead_y2 = y1 + int(height * 0.2)
```

```
            # Центр лба (середина по X и Y)
            forehead_center_x = (x1 + x2) // 2
            forehead_center_y = (forehead_y1 + forehead_y2) // 2

            # Отрисовка
            cv2.rectangle(processed_frame, (x1, y1), (x2, y2), (0, 255, 0), 1)    # Голова
            cv2.rectangle(processed_frame, (x1, forehead_y1), (x2, forehead_y2), (0, 0, 255), 2)
            cv2.circle(processed_frame, (forehead_center_x, forehead_center_y), 7, (255, 0, 0), -1)

            # Вывод соотношения сторон для отладки
            cv2.putText(processed_frame, f"AR: {aspect_ratio:.2f}", (x1, y1 - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 0), 1)

    # FPS-счётчик (для оценки производительности)
    fps = cap.get(cv2.CAP_PROP_FPS)
    cv2.putText(processed_frame, f"FPS: {fps:.1f}", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 255), 2)

    cv2.imshow("Turret Targeting", processed_frame)
    if cv2.waitKey(1) == 27:  # Выход по ESC
        break

cap.release()
cv2.destroyAllWindows()
```