МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №4

Специальность ИИ(з)

Выполнил
А.Ю. Кураш,
студент группы ИИ-24

Проверил
Андренко К.В,
Преподаватель-стажер кафедры ИИТ,
«___k _____2025 г.

Брест 2025

Цель: исследовать применение алгоритмов трекинга на базе обученной сети-детектора объектов

Общее задание

1. Используя сеть-детектор, обученный в ЛР 3, реализовать логику для отслеживания множественных объектов, используя библиотеку Ultralytics YOLO;
2. Применять алгоритмы BoT-Sort и ByteTrack (задействовать соответствующие конфигурационные файлы);
3. Исследовать изменения параметров в конфигурационных файлах и их влияние на качество трекинга;
4. В качестве исходных видеоматериалов для экспериментов использовать видео-ролики из сети (например, из YouTube), содержащие множественные объекты классов из ЛР 3;
5. Оформить отчет по выполненной работе, залить исходный код и отчет в соответствующий репозиторий на github.

8	YOLOv11n	**Номерные знаки авто:** https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e/dataset/11

Выполнение:

Код программы

```
import os
import cv2
import tkinter as tk
from tkinter import filedialog, ttk
from PIL import Image, ImageTk
from ultralytics import YOLO

class LicensePlateTrackerApp:
    def __init__(self, master):
        self.master = master
        master.title("YOLOv11 Video Tracker (BoT-SORT & ByteTrack)")
        master.geometry("1000x700")

        # --- НАСТРОЙКИ МОДЕЛИ ---
        # Укажите путь к вашей модели
        self.model_path = "C:\\Users\\User\\OneDrive\\Desktop\\IP_AI_24\\reports\\Kurash\\lab3\\src\\yolov11n.pt" # Или ваш путь к best.pt

        try:
            self.model = YOLO(self.model_path)
            print(f"Модель загружена: {self.model_path}")
        except Exception as e:
            print(f"Ошибка загрузки модели: {e}")
            master.destroy()
            return
```

```
        self.video_path = None
        self.cap = None
        self.is_running = False

        # --- ЭЛЕМЕНТЫ GUI ---

        # 1. Панель управления (верхняя часть)
        control_frame = tk.Frame(master)
        control_frame.pack(side=tk.TOP, fill=tk.X, padx=10, pady=10)

        # Кнопка выбора видео
        self.btn_select = tk.Button(control_frame, text="1. Выбрать видео", command=self.select_video)
        self.btn_select.pack(side=tk.LEFT, padx=5)

        # Выпадающий список выбора трекера
        tk.Label(control_frame, text="2. Алгоритм трекинга:").pack(side=tk.LEFT, padx=5)
        self.tracker_selector = ttk.Combobox(control_frame, values=["botsort.yaml", "bytetrack.yaml"])
        self.tracker_selector.current(0) # По умолчанию BoT-SORT
        self.tracker_selector.pack(side=tk.LEFT, padx=5)

        # Кнопка старт/стоп
        self.btn_start = tk.Button(control_frame, text="3. Старт/Стоп", command=self.toggle_tracking, state=tk.DISABLED)
        self.btn_start.pack(side=tk.LEFT, padx=5)

        # Статус бар
        self.lbl_status = tk.Label(master, text="Готов к работе", bd=1, relief=tk.SUNKEN, anchor=tk.W)
        self.lbl_status.pack(side=tk.BOTTOM, fill=tk.X)

        # 2. Область отображения видео
        self.canvas_frame = tk.Frame(master, bg="black")
        self.canvas_frame.pack(expand=True, fill=tk.BOTH)

        self.canvas = tk.Canvas(self.canvas_frame, bg="gray")
        self.canvas.pack(expand=True, fill=tk.BOTH)

    def select_video(self):
        """Открывает диалог выбора видеофайла."""
        self.video_path = filedialog.askopenfilename(
            title="Выберите видеофайл",
            filetypes=(("Video files", "*.mp4 *.avi *.mkv *.mov"), ("All files", "*.*"))
        )
        if self.video_path:
            self.lbl_status.config(text=f"Выбрано видео: {os.path.basename(self.video_path)}")
            self.btn_start.config(state=tk.NORMAL)
            # Сброс предыдущего видео, если было
            if self.cap:
                self.cap.release()
            self.cap = cv2.VideoCapture(self.video_path)

            # Показать первый кадр для превью
            ret, frame = self.cap.read()
            if ret:
                self.display_frame(frame)
                self.cap.set(cv2.CAP_PROP_POS_FRAMES, 0) # Вернуться в начало

    def toggle_tracking(self):
        """Переключает состояние воспроизведения/трекинга."""
        if not self.is_running:
            self.is_running = True
            self.btn_start.config(text="Стоп")
            self.tracker_config = self.tracker_selector.get()
            self.lbl_status.config(text=f"Трекинг запущен: {self.tracker_config}")
            self.process_video_loop()
        else:
            self.is_running = False
            self.btn_start.config(text="Старт")
            self.lbl_status.config(text="Пауза")

    def process_video_loop(self):
```

```
        """Основной цикл обработки видеокадров."""
        if not self.is_running or not self.cap.isOpened():
            return

        ret, frame = self.cap.read()
        if not ret:
            self.is_running = False
            self.btn_start.config(text="Старт (Видео завершено)")
            self.lbl_status.config(text="Видео завершено")
            self.cap.set(cv2.CAP_PROP_POS_FRAMES, 0) # Перемотка в начало
            return

        # --- ГЛАВНАЯ ЧАСТЬ: TRECKING ---
        try:
            # Используем model.track вместо model.predict
            # persist=True важен для сохранения ID объектов между кадрами
            results = self.model.track(
                source=frame,
                persist=True,
                tracker=self.tracker_config, # 'botsort.yaml' или 'bytetrack.yaml'
                verbose=False # Убрать лишний вывод в консоль
            )

            # Отрисовка результатов (plot() рисует боксы и ID треков)
            annotated_frame = results[0].plot()

            # Отображение
            self.display_frame(annotated_frame)

        except Exception as e:
            print(f"Ошибка трекинга: {e}")
            self.is_running = False
            return

        # Запланировать следующий кадр через ~30 мс (около 30 FPS)
        self.master.after(10, self.process_video_loop)

    def display_frame(self, frame_bgr):
        """Конвертирует кадр OpenCV и выводит на Canvas."""
        # BGR -> RGB
        frame_rgb = cv2.cvtColor(frame_bgr, cv2.COLOR_BGR2RGB)
        img_pil = Image.fromarray(frame_rgb)

        # Масштабирование под размер окна
        cw = self.canvas.winfo_width()
        ch = self.canvas.winfo_height()

        # Защита от нулевых размеров при старте
        if cw < 10 or ch < 10: cw, ch = 800, 600

        # Сохраняем пропорции
        img_pil.thumbnail((cw, ch), Image.LANCZOS)

        self.tk_image = ImageTk.PhotoImage(img_pil)
        self.canvas.delete("all")
        self.canvas.create_image(cw//2, ch//2, anchor=tk.CENTER, image=self.tk_image)

if __name__ == "__main__":
    root = tk.Tk()
    app = LicensePlateTrackerApp(root)
    root.mainloop()
```

Вывод: Я изучил работу с моделями YOLO и научился настраивать их для определенных задач.