

Министерство образования Республики Беларусь  
Учреждение образования  
«Брестский государственный технический университет»  
Кафедра ИИТ

Лабораторная работа №4  
По дисциплине: «Обработка изображений в интеллектуальных системах»  
Тема: «Трекинг множественных объектов»

Выполнил:  
Студент 4 курса  
Группы ИИ-24  
Капуза Н.А.  
Проверила:  
Андренко К.В.

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.

Ход работы:

Код программы:

```
import cv2
from ultralytics import YOLO

def run_tracking(model_path, video_path, tracker_config_file, output_name):
    """
    Функция для запуска трекинга с определенным алгоритмом.
    """
    print(f"--- Запуск трекинга с конфигом: {tracker_config_file} ---")

    # 1. Загрузка модели (Используем Нашу обученную модель из ЛР 3)
    model = YOLO(model_path)

    # 2. Запуск трекинга
    # persist=True важен для сохранения ID объектов между кадрами
    results = model.track(
        source=video_path,
        tracker=tracker_config_file,
        conf=0.5,    # Порог уверенности (можно менять для пункта 3)
        iou=0.5,     # Порог IOU (можно менять для пункта 3)
```

```

    show=True, # Показывать видео в реальном времени
    save=True, # Сохранить результат
    project='runs/track', # Папка для сохранения
    name=output_name, # Имя подпапки
    persist=True # Важно для трекинга!
)

print(f"Готово. Результаты сохранены в runs/track/{output_name}")

if __name__ == "__main__":
    # ПУТЬ К НАШЕЙ МОДЕЛИ ИЗ ЛР 3
    MY_MODEL = r"D:\4 Курс\Oiiiis\runs\detect\cat_yolov10_result_win\weights\best.pt"

    # ПУТЬ К ВИДЕО
    MY_VIDEO = r"C:\Users\kolya\Downloads\cat.mp4"

    # Эксперимент 1: Использование BoT-SORT
    # botsort.yaml - это встроенный конфиг в ultralytics
    run_tracking(MY_MODEL, MY_VIDEO, "botsort.yaml", "experiment_botsort")

    # Эксперимент 2: Использование ByteTrack
    # bytetrack.yaml - это встроенный конфиг в ultralytics
    run_tracking(MY_MODEL, MY_VIDEO, "bytetrack.yaml", "experiment_bytetrack")

```

Результат:

BoT-SORT:



ByteTrack:



## Разница алгоритмов

1. BoT-SORT: Лучше работает, когда объекты движутся, и камера тоже движется. Он использует признаки внешнего вида (ReID) для восстановления потерянных треков. Обычно точнее, но чуть медленнее.
2. ByteTrack: Очень быстрый алгоритм. Он пытается связать даже те объекты, которые детектор распознал с низкой уверенностью. Хорош, если объекты перекрывают друг друга, но может давать больше ошибок (путать объекты).

Вывод: исследовал применение алгоритмов трекинга на базе обученной сети-детектора объектов.