Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра интеллектуально-информационных технологий

# Обработка изображений в ИС
## Лабораторная работа №4
## Трекинг множественных объектов

Выполнила:
студентка 4 курса
группы ИИ-24
Алешко А. В.
Проверила:
Андренко К. В.

Брест-2025

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

**Общее задание:**

1. 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..

| В-т | Детектор | Датасет |
|---|---|---|
| 1 | YOLOv10n | Люди: https://universe.roboflow.com/leo-ueno/people-detection-o4rdr/dataset/10 |

**Код программы:**

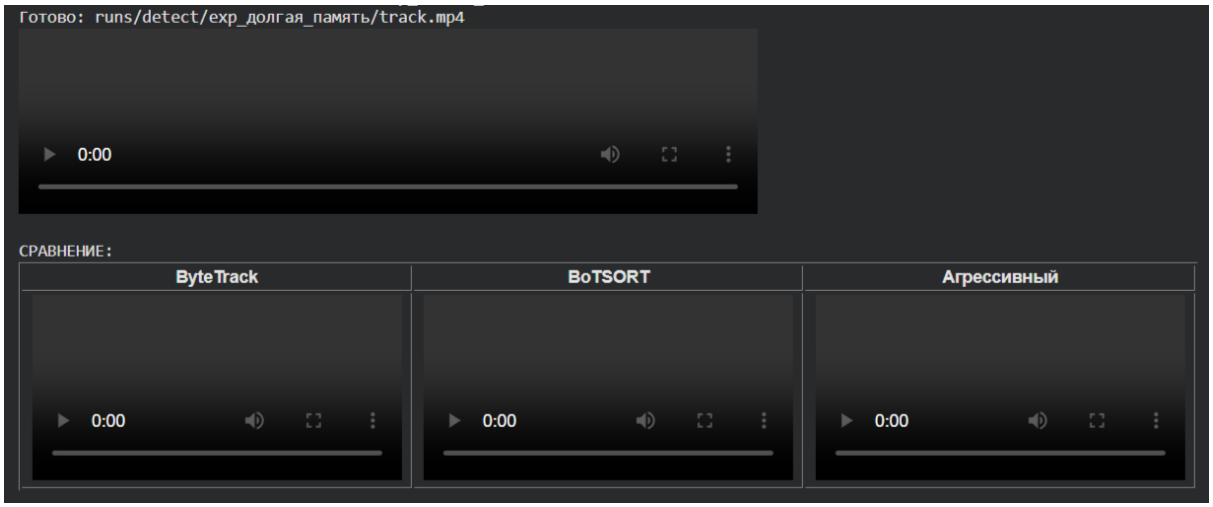
```
import os
import cv2
import numpy as np
from pathlib import Path
from ultralytics import YOLO
import supervision as sv
from IPython.display import Video, HTML, display

print("Зависимости установлены.")
#Загрузка модели best.pt
model_path = "best.pt"
if not os.path.exists(model_path):
    possible =
"/content/runs/detect/yolov10n_people_detection/weights/best.p
t"
    if os.path.exists(possible):
        model_path = possible
    else:
        print("ЗАГРУЗИТЕ МОДЕЛЬ best.pt:")
```

```
        from google.colab import files
        uploaded = files.upload()
        model_path = list(uploaded.keys())[0]
model = YOLO(model_path)
model.fuse()
print(f"Модель загружена: {model_path}")

video_path = "street3.mp4"
cap = cv2.VideoCapture(video_path)
w = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
h = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = cap.get(cv2.CAP_PROP_FPS)
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
cap.release()
print(f"Видео: {w}x{h}, FPS: {fps:.1f}, кадров:
{total_frames}")

box_annotator = sv.BoxCornerAnnotator(thickness=2)
label_annotator = sv.LabelAnnotator(text_scale=0.6,
text_thickness=1)

def annotate_frame(frame, results):
    detections = sv.Detections.from_ultralytics(results)
    annotated_frame = frame.copy()
    annotated_frame = box_annotator.annotate(annotated_frame,
detections)
    labels = [f"ID:{int(id)} {conf:.2f}" for id, conf in
zip(results.boxes.id, results.boxes.conf)]
    annotated_frame =
label_annotator.annotate(annotated_frame, detections,
labels=labels)
    return annotated_frame

#ByteTrack
print("\nЗапуск ByteTrack...")
conf_path = "bytetrack.yaml"
if not os.path.exists(conf_path):
    !wget -q
https://raw.githubusercontent.com/ultralytics/ultralytics/main
/ultralytics/cfg/trackers/bytetrack.yaml -O {conf_path}
model.track(
    source=video_path,
    conf=0.25,
    iou=0.45,
    persist=True,
    tracker=conf_path,
    save=True,
    name="bytetrack",
    exist_ok=True
)
```

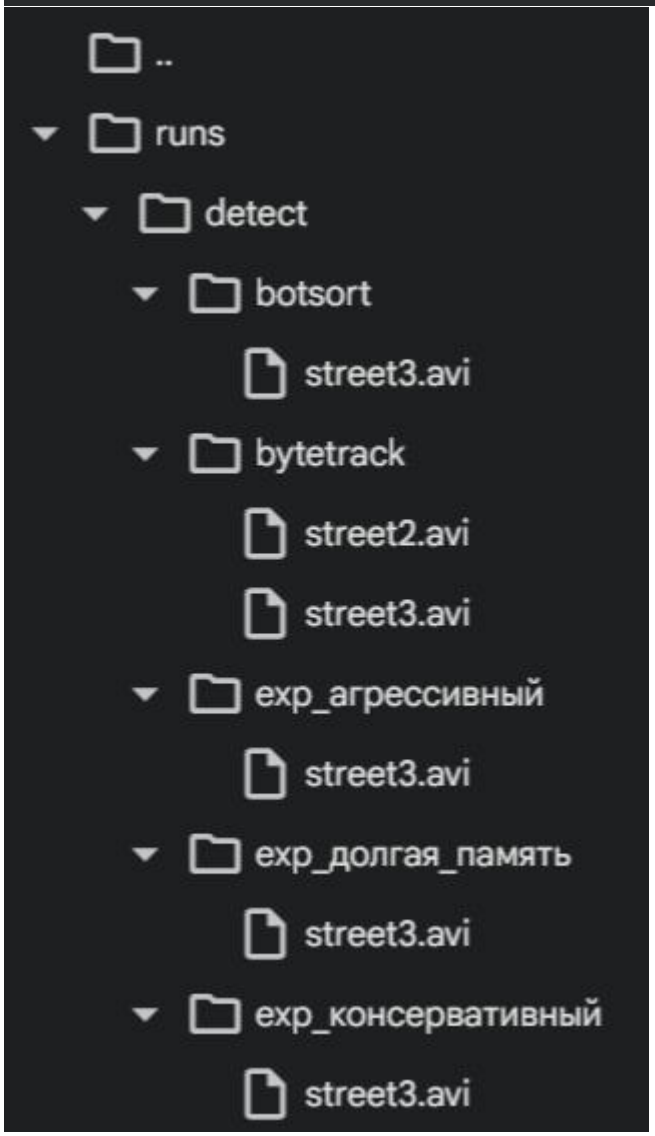
```
output_byte = "runs/detect/bytetrack/track.mp4"
print("ByteTrack готов!")
display(Video(output_byte, embed=True, width=800))  #
embed=True!

#BoTSORT
print("\nЗапуск BoTSORT...")
conf_path = "botsort.yaml"
if not os.path.exists(conf_path):
    !wget -q
https://raw.githubusercontent.com/ultralytics/ultralytics/main
/ultralytics/cfg/trackers/botsort.yaml -O {conf_path}
model.track(
    source=video_path,
    conf=0.25,
    iou=0.45,
    persist=True,
    tracker=conf_path,
    save=True,
    name="botsort",
    exist_ok=True
)

output_bot = "runs/detect/botsort/track.mp4"
print("BoTSORT готов!")
display(Video(output_bot, embed=True, width=800))

print("\nЭксперименты с параметрами:")
experiments = [
    {"name": "агрессивный", "conf": 0.1, "iou": 0.7,
"tracker": "bytetrack.yaml"},
    {"name": "консервативный", "conf": 0.5, "iou": 0.3,
"tracker": "botsort.yaml"},
    {"name": "долгая_память", "conf": 0.25, "iou": 0.45,
"tracker": "botsort.yaml"},
]
for exp in experiments:
    print(f"\n--- {exp['name']} ---")
    cfg = exp["tracker"]
    if not os.path.exists(cfg):
        !wget -q
https://raw.githubusercontent.com/ultralytics/ultralytics/main
/ultralytics/cfg/trackers/{cfg} -O {cfg}
    model.track(
        source=video_path,
        conf=exp["conf"],
        iou=exp["iou"],
        persist=True,
        tracker=cfg,
        save=True,
        name=f"exp_{exp['name']}",
```

```
        exist_ok=True
    )
    out_path = f"runs/detect/exp_{exp['name']}/track.mp4"
    print(f"Готово: {out_path}")
    display(Video(out_path, embed=True, width=600))

#Сравнение
print("\nСРАВНЕНИЕ:")
display(HTML(f"""
<table border="1" style="width:100%; text-align:center;">

<tr><th>ByteTrack</th><th>BoTSORT</th><th>Агрессивный</th></tr
>
  <tr>
    <td><video src="{output_byte}" width=300
controls></video></td>
    <td><video src="{output_bot}" width=300
controls></video></td>
    <td><video src="runs/detect/exp_агрессивный/track.mp4"
width=300 controls></video></td>
  </tr>
</table>
"""))
```

Результат работы программы:
1. ByteTrack: Быстрый, подходит для реального времени, но теряет ID при окклюзиях.
2. BoTSORT: Точнее благодаря ReID, лучше для сложных сцен.
3. Параметры:
   • conf ↓ → больше детекций/треков (агрессивно)
   • iou ↑ → строже matching, меньше ложных ID
   • persist=True → держит ID между кадрами.



```
video 1/1 (frame 2916/2935) /content/street3.mp4: 384x640 7 persons, 126.8ms
video 1/1 (frame 2917/2935) /content/street3.mp4: 384x640 7 persons, 122.6ms
video 1/1 (frame 2918/2935) /content/street3.mp4: 384x640 6 persons, 128.2ms
video 1/1 (frame 2919/2935) /content/street3.mp4: 384x640 5 persons, 126.2ms
video 1/1 (frame 2920/2935) /content/street3.mp4: 384x640 6 persons, 138.2ms
video 1/1 (frame 2921/2935) /content/street3.mp4: 384x640 6 persons, 125.6ms
video 1/1 (frame 2922/2935) /content/street3.mp4: 384x640 7 persons, 132.0ms
video 1/1 (frame 2923/2935) /content/street3.mp4: 384x640 8 persons, 125.7ms
video 1/1 (frame 2924/2935) /content/street3.mp4: 384x640 8 persons, 124.1ms
video 1/1 (frame 2925/2935) /content/street3.mp4: 384x640 8 persons, 128.5ms
video 1/1 (frame 2926/2935) /content/street3.mp4: 384x640 7 persons, 123.3ms
video 1/1 (frame 2927/2935) /content/street3.mp4: 384x640 8 persons, 134.3ms
video 1/1 (frame 2928/2935) /content/street3.mp4: 384x640 8 persons, 128.4ms
video 1/1 (frame 2929/2935) /content/street3.mp4: 384x640 7 persons, 137.5ms
video 1/1 (frame 2930/2935) /content/street3.mp4: 384x640 7 persons, 124.3ms
video 1/1 (frame 2931/2935) /content/street3.mp4: 384x640 8 persons, 125.5ms
video 1/1 (frame 2932/2935) /content/street3.mp4: 384x640 8 persons, 126.0ms
video 1/1 (frame 2933/2935) /content/street3.mp4: 384x640 8 persons, 125.2ms
video 1/1 (frame 2934/2935) /content/street3.mp4: 384x640 8 persons, 123.4ms
video 1/1 (frame 2935/2935) /content/street3.mp4: 384x640 7 persons, 142.2ms
```




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