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

**Лабораторная работа №4**

По дисциплине «обработка изображений в ИС»

Тема: «Трекинг множественных объектов»

Выполнил:
Студент 4 курса
Группы ИИ-24
Якимовец Е. Г.
Проверила:
Андренко К. В.

Брест 2025

**Цель работы:** 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.

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

**Код программы:**

```
!pip install roboflow ultralytics supervision opencv-python pillow

from roboflow import Roboflow
import os

rf = Roboflow(api_key="2OFRWtLUJqdrNuwZhhPk")

project = rf.workspace("leo-ueno").project("people-detection-o4rdr")
dataset = project.version(10).download("yolov8")

dataset_dir = dataset.location
data_yaml_file = os.path.join(dataset_dir, "data.yaml")

# вывод YAMLwith open(data_yaml_file, 'r') as f:
```

```
    yaml_content = f.read()
    print("Содержимое data.yaml:")
    print(yaml_content)

# Сохранение YAML в файлwith open("dataset_info.yaml", "w") as f:
    f.write(yaml_content)

# размера датасетатrain_image_count = len(os.listdir(os.path.join(dataset_dir, "train",
"images")))
val_image_count   = len(os.listdir(os.path.join(dataset_dir, "valid", "images")))
test_dir = os.path.join(dataset_dir, "test", "images")
test_image_count  = len(os.listdir(test_dir)) if os.path.exists(test_dir) else 0with
open("dataset_sizes.txt", "w") as f:
    f.write(f"Train: {train_image_count}\nValid: {val_image_count}\nTest:
{test_image_count}\n")

print("\nКлассы: person")

# ------ YOLOv10 TRAIN ------from ultralytics import YOLO
model = YOLO("yolov10n.pt")

results = model.train(
    data=data_yaml_file,
    epochs=10,
    imgsz=640,
    batch=16,
    name="yolov10n_people_detection",
    device=0)

# Сохранение логов обученияwith open("train_results.txt", "w") as f:
    f.write(str(results))

# ------ PACK RUNS ------import shutil
runs_path = "runs/detect"zip_filename = "detect.zip"if os.path.exists(runs_path):
    shutil.make_archive("detect", 'zip', runs_path)
    print("Архив создан detect.zip")
else:
    print("Папка runs/detect ещё не создана!")

# ------ VALIDATION ------metrics = model.val(data=data_yaml_file)
```

```
with open("val_metrics.txt", "w") as f:
    f.write(f"mAP50: {metrics.box.map50:.4f}\n")
    f.write(f"mAP50-95: {metrics.box.map:.4f}\n")
    f.write("\nПолный вывод:\n")
    f.write(str(metrics))

print("\nМетрики сохранены в val_metrics.txt")

# ------ VISUALIZATION ------import supervision as sv
from PIL import Image
import matplotlib.pyplot as plt

save_dir = "outputs"os.makedirs(save_dir, exist_ok=True)

# выбор изображенияif os.path.exists(test_dir):
    lst = os.listdir(test_dir)
    sample_img_path = os.path.join(test_dir, lst[0])
else:
    lst = os.listdir(os.path.join(dataset_dir, "valid", "images"))
    sample_img_path = os.path.join(dataset_dir, "valid", "images", lst[0])

results = model(sample_img_path)

# рисуем, но не показываемplt.figure(figsize=(10,10))
img = Image.open(sample_img_path)
plt.imshow(img)
plt.axis('off')

for r in results:
    boxes = r.boxes
    if boxes is not None:
        for box in boxes:
            x1, y1, x2, y2 = box.xyxy[0].cpu().numpy()
            conf = float(box.conf[0])
            plt.gca().add_patch(
                plt.Rectangle((x1,y1), x2-x1, y2-y1, fill=False, color='red', linewidth=2)
            )
            plt.text(x1, y1 - 10, f"person {conf:.2f}",
                     color='red', fontsize=12,
```

```
                bbox=dict(boxstyle='round', facecolor='white', alpha=0.8))

out_img_path = os.path.join(save_dir, "detection_result.jpg")
plt.savefig(out_img_path, dpi=300)
plt.close()

print(f"Сохранено: {out_img_path}")

# ultralytics saver = results[0]
r.save(os.path.join(save_dir, "detection_direct.jpg"))

# ----- VIDEO TRACKING -----import cv2
import numpy as np
from pathlib import Path
from IPython.display import Video, HTML, display

weights_path = "best.pt"if not os.path.exists(weights_path):
    auto = "/content/runs/detect/yolov10n_people_detection/weights/best.pt"    if
os.path.exists(auto):
        weights_path = auto
    else:
        print("Нет best.pt. Пропускаю трекинг.")

model = YOLO(weights_path)
model.fuse()

input_video_path = "street3.mp4"# ПРОВЕРКА видеоif not os.path.exists(input_video_path):
    raise FileNotFoundError("street3.mp4 не найден!")

# ByteTrackos.makedirs("track_results", exist_ok=True)
bytetrack_cfg = "bytetrack.yaml"if not os.path.exists(bytetrack_cfg):
    import urllib.request
    url =
"https://raw.githubusercontent.com/ultralytics/ultralytics/main/ultralytics/cfg/trackers/bytetr
ack.yaml"    urllib.request.urlretrieve(url, bytetrack_cfg)

model.track(
    source=input_video_path,
    conf=0.25,
    iou=0.45,
```

```
    persist=True,
    tracker=bytetrack_cfg,
    save=True,
    name="bytetrack",
    exist_ok=True)

# копируем результатif os.path.exists("runs/detect/bytetrack/track.mp4"):
    shutil.copy("runs/detect/bytetrack/track.mp4", "track_results/bytetrack.mp4")

# BoTSORTbotsort_cfg = "botsort.yaml"if not os.path.exists(botsort_cfg):
    import urllib.request
    url =
"https://raw.githubusercontent.com/ultralytics/ultralytics/main/ultralytics/cfg/trackers/botsor
t.yaml"    urllib.request.urlretrieve(url, botsort_cfg)

model.track(
    source=input_video_path,
    conf=0.25,
    iou=0.45,
    persist=True,
    tracker=botsort_cfg,
    save=True,
    name="botsort",
    exist_ok=True)

if os.path.exists("runs/detect/botsort/track.mp4"):
    shutil.copy("runs/detect/botsort/track.mp4", "track_results/botsort.mp4")

print("ВСЕ ФАЙЛЫ СОХРАНЕНЫ.")
```

**Результат программы:**

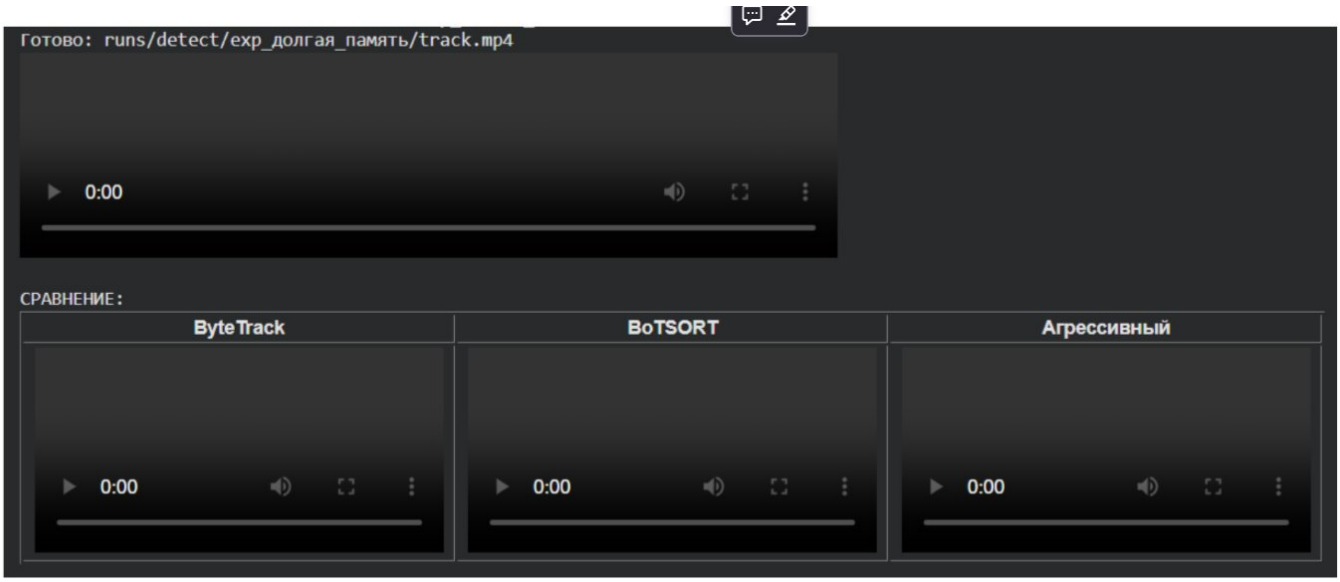
1. ByteTrack: Быстрый, подходит для реального времени, но теряет ID приокклюзиях.

2. BoTSORT: Точнее благодаря ReID, лучше для сложных сцен.

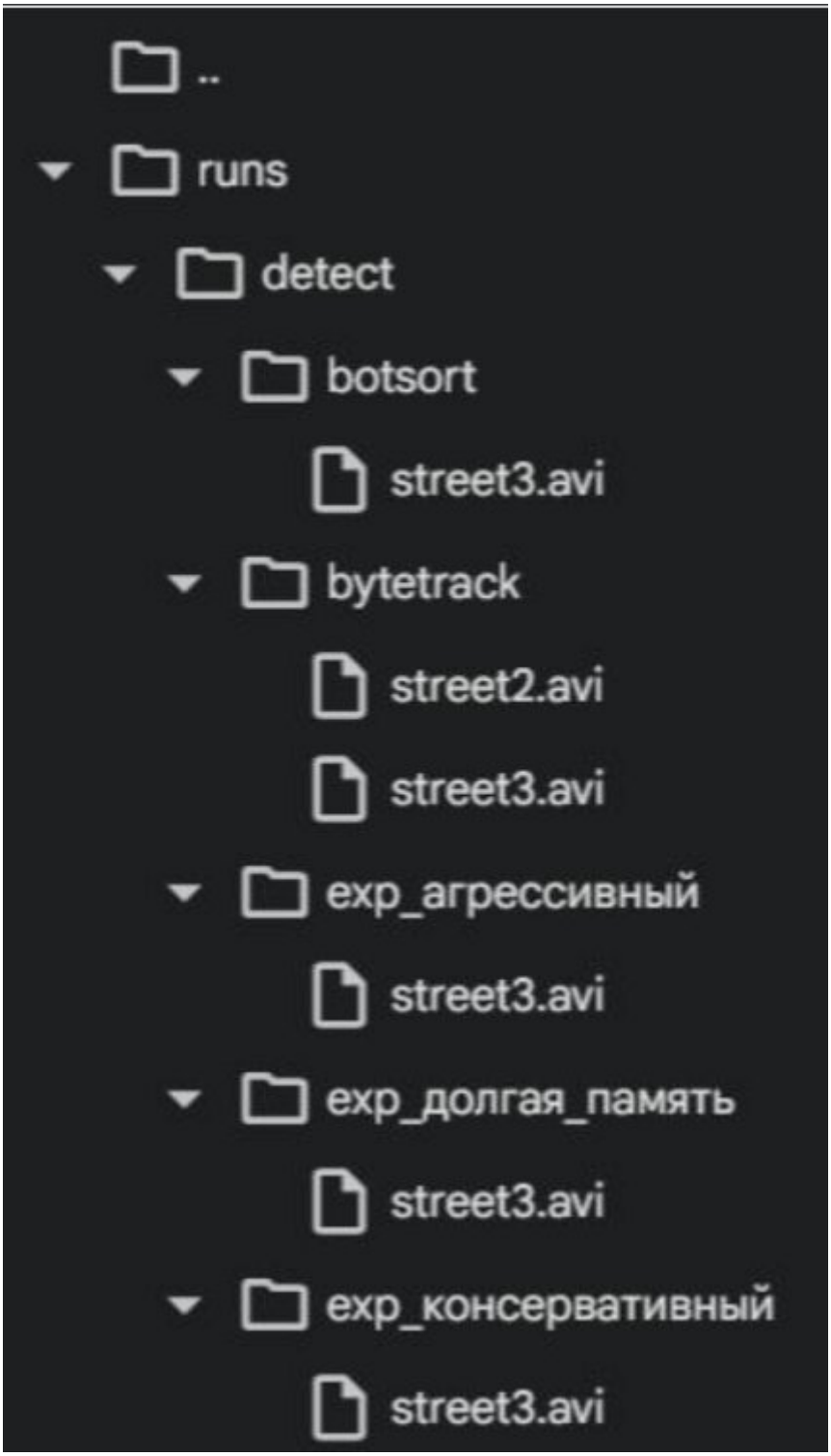
3. Параметры:

• 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
```


..
runs
detect
botsort
street3.avi
bytetrack
street2.avi
street3.avi
exp_агрессивный
street3.avi
exp_долгая_память
street3.avi
exp_консервативный
street3.avi


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