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

Лабораторная работа №4

По дисциплине: «ОИвИС»

Тема: «Трекинг множественных объектов»

Выполнил:
Студент 4 курса
Группы ИИ-23
Романюк А. П.
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.Вариант:

Код программы:

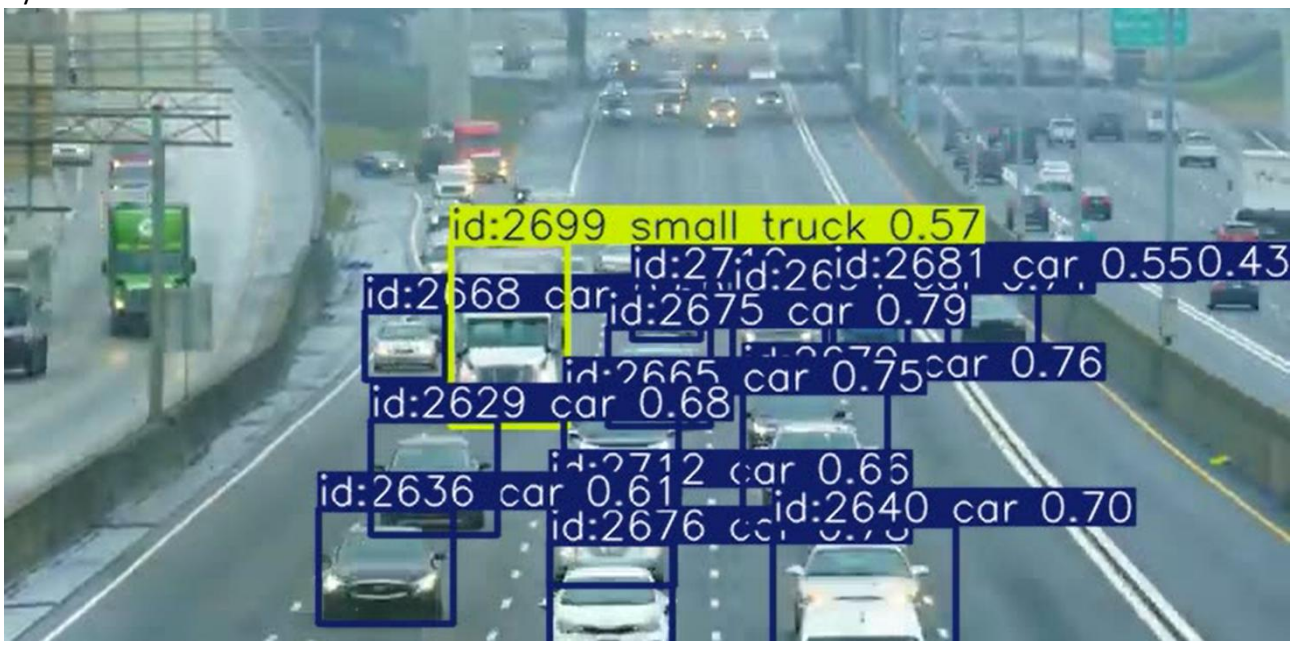
```
!pip install ultralytics supervision opencv-python -q

from ultralytics import YOLO

model = YOLO("best.pt")

model.track(
   source="video.mp4",
   conf=0.25,
   iou=0.45,
   tracker="botsort.yaml",
   show=False,
   save=True,
   name="botsort"
)

model.track(
   source="video.mp4",
   conf=0.25,
   iou=0.45,
   tracker="bytetrack.yaml",
   show=False,
   save=True,
   name="bytetrack"
```

)

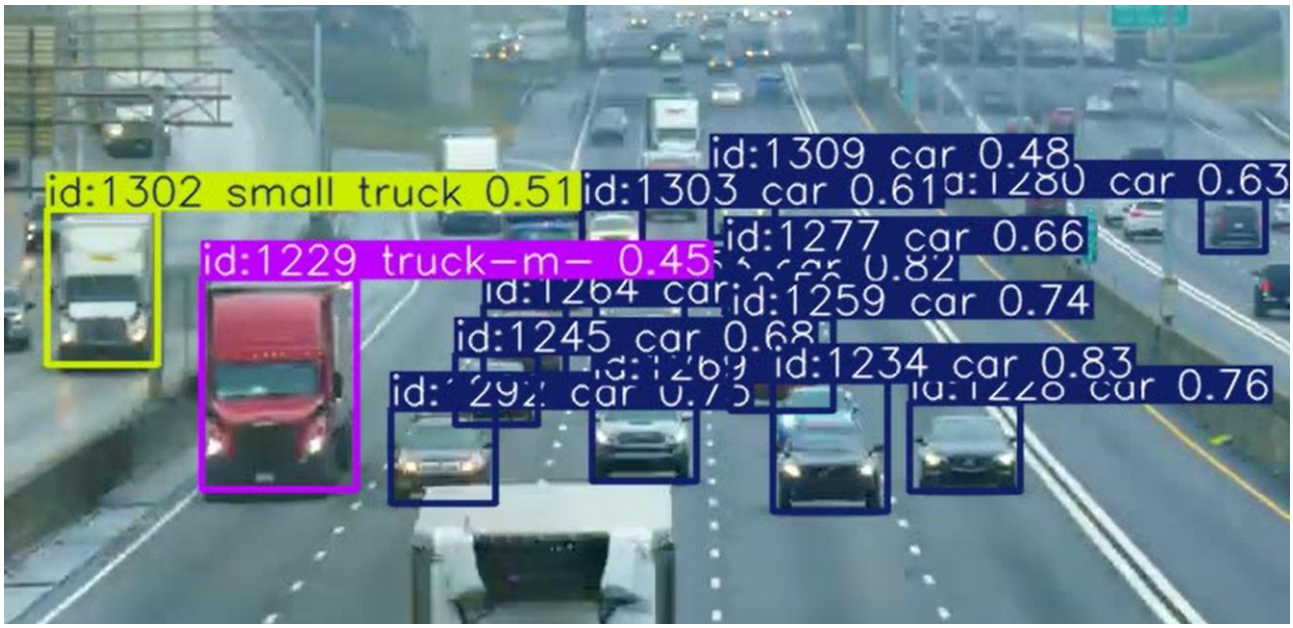
!zip -r videos.zip /content/runs/detect/

from google.colab import files
files.download('videos.zip')

Bytesort:



Botsort:



В ходе экспериментов с алгоритмами ByteTrack и BoT-SORT было проведено изменение нескольких ключевых параметров, что позволило оценить их влияние на устойчивость и качество трекинга при использовании обученной в ЛР3 модели-детектора.

При работе с ByteTrack наиболее заметное влияние оказал параметр **track_thresh**, определяющий порог уверенности детекции, которая допускается к дальнейшему трекингу. При снижении этого порога трекер начинал лучше «цепляться» за объекты на дальних расстояниях и частично закрытые объекты, однако одновременно увеличивалось количество ложных треков, которые появлялись на фоне шумовых детекций. Напротив, повышение порога делало трекинг чище, но увеличивало количество ситуаций, когда объект

терялся. Параметр **match_thresh**, контролирующий минимальный IoU для сопоставления текущих детекций с существующими треками, тоже сильно влиял на стабильность ID: при низких значениях ID-switch случались чаще, а при более высоких трекер становился устойчивее, но хуже работал при резком движении объектов. Изменение **track_buffer** также показало важное влияние: большие значения позволяли удерживать трек даже при временных пропусках детектора, однако иногда это приводило к появлению «фантомных» треков, которые продолжали существовать, даже когда объект уже давно покинул кадр.

BoT-SORT продемонстрировал совершенно другой характер поведения. В данном алгоритме качество напрямую зависело от точности работы детектора: изменение **det_thresh** приводило к большим колебаниям качества. Низкие пороги вызывали появление большого числа ошибочных траекторий, поскольку BoT-SORT очень чувствителен к детекционному шуму. Увеличение этого порога улучшало качество — но трекер начинал терять объекты, даже если они кратковременно становились менее уверенно распознаваемыми. Важную роль сыграл и параметр **conf_thres**, определяющий, какие детекции передаются в модуль ReID. Оптимальным оказалось его значение на уровне порядка 0.4–0.5: при меньших значениях ReID начинал подбирать неверные визуальные признаки, при больших — просто не получал достаточно информации.

Особенно заметным оказалось влияние **ReID-модуля** — включение параметра «with_reid» сильно улучшало устойчивость траекторий. С ним BoT-SORT почти полностью переставал путать объекты между собой даже при пересечении траекторий. Без ReID метод начинал вести себя почти как классический SORT, с частыми потерями идентификаторов. Параметр **gamma**, регулирующий поведение фильтра Калмана, определял степень сглаживания траекторий: увеличение значения делало движения объектов более плавными, но ухудшало реакцию на резкие изменения направления, из-за чего иногда появлялись внезапные смещения трека.

В сравнении двух методов изменения параметров проявили противоположный характер: ByteTrack оказался гораздо более устойчивым к настройкам и уверенно работал даже при довольно агрессивном снижении порогов, в то время как BoT-SORT требовал точной подстройки и сильно зависел от качества детекций. В целом ByteTrack лучше показал себя в сложных и динамичных сценах, а BoT-SORT — при необходимости максимально точного ведения отдельных объектов, особенно если используется ReID.

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