Autók számlálása YOLO és DeepSort segítségével

A YOLO modell alapértelmezetten 80 különböző objektumot képes felismerni, melyek között megtalálható a személyautó, a busz és a teherautó is, ezért a feladat megoldásához tökéletesen megfelel. A DeepSort a felismert járművek nyomon követeséhez szükséges, cv2 pedig a video megjelenítéséhez.

```
from ultralytics import YOLO
from deep_sort_realtime.deepsort_tracker import DeepSort
import cv2

model = YOLO("yolov8n.pt")

tracker = DeepSort(max_age=15, n_init=3, max_iou_distance=0.7)
```

importáljuk be a modelleket

A DeepSort paraméterei között látható a „max_age”, ez azt jelenti, hogy hány képkockán keresztül „nem él” a korábban felismert bounding box, mielőtt eldobja. Az „n_init” pedig azt jelenti, hogy hány képkockán keresztül kell léteznie ahhoz, hogy számba vegye. Az „iou” azt jelenti, hogy mennyi átfedésnek kell lennie a predikció és a valós érték között.

```
cap = cv2.VideoCapture("video3.mp4")

line_y = 700

count = 0

prev_positions = {}
```

A feladat megoldásához szükséges változók.

```
cv2.namedWindow("YOLO+DeepSort - Autószámlálás", cv2.WINDOW_NORMAL)
```

A cv2 ablak

```
while True:
    ret, frame = cap.read()
    if not ret:
        break
```

Ez egy olyan ciklus, ami addig ismétlődik, ameddig a video tart.

```
results = model(frame, verbose=False)
dets = []

for r in results:
    for box in r.boxes:
        cls = int(box.cls[0])

        # car=2, bus=5, truck=7 (COCO osztályok)
        if cls in [2, 5, 7]:
            # tensor -> float lista
            x1, y1, x2, y2 = box.xyxy[0].tolist()
            w = x2 - x1
            h = y2 - y1
            conf = float(box.conf[0])
            # DeepSort formátum: [[x, y, w, h], conf, class_id]
            dets.append([[x1, y1, w, h], conf, cls])
```

A YOLO képkockánként vizsgálja a videot, ha a detektált objektum(ok) megegyeznek a meghatározott osztályokkal, akkor azpkat elmenti a „dets” listába. A tenzorból kinyert adatok átalakítása nagyon fontos, mert a DeepSort csak ebben a formátumban fogadja el.

```
tracks = tracker.update_tracks(dets, frame=frame)

for track in tracks:
    if not track.is_confirmed():
        continue

    x, y, w, h = track.to_ltwh()
    cx = x + w / 2
    cy = y
    track_id = track.track_id
```

A detektált objektumok kapnak új track id-t.

```
if track_id not in prev_positions:
    prev_positions[track_id] = cy

prev_y = prev_positions[track_id]
```

Ha eddig nem létezett az id a szótárban, akkor létrehozza.

```
#vonalátlépés
if prev_y > line_y >= cy:
    count += 1
    print(f">>> AUTÓ SZÁMLÁLVA, ID={track_id}, összesen={count}")
```

Akkor lépte át a vonalat ha a számláló vonal a jelenlegi és a előző pozíciója között van.

```
cv2.rectangle(frame,
              (int(x), int(y)),
              (int(x + w), int(y + h)),
              (0, 255, 0), 2)
cv2.putText(frame, f"ID {track_id}",
            (int(x), int(y - 5)),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.5, (0, 255, 0), 1)
```

Rajzoljuk ki a bounding boxot.

```
# Számlálóvonal kirajzolása
cv2.line(frame, (0, line_y),
         (frame.shape[1], line_y),
         (255, 0, 0), 2)
```

Számlálóvonal kirajzolása.

```
# Számláló kiírása
cv2.putText(frame, f"Count: {count}", (20, 50),
            cv2.FONT_HERSHEY_SIMPLEX,
            1, (0, 0, 255), 2)
resized = cv2.resize(frame, (580,750), interpolation=cv2.INTER_AREA)
```

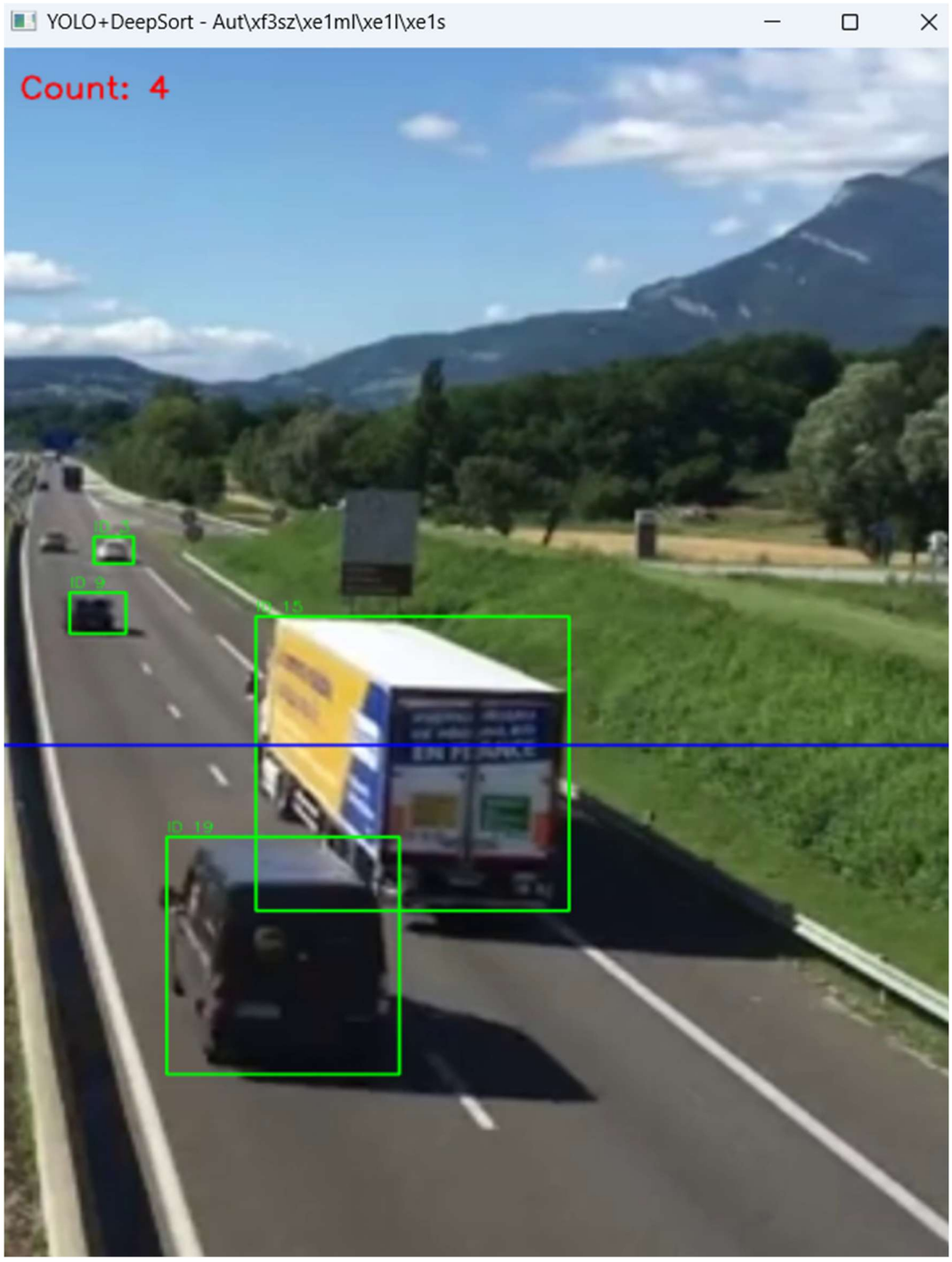
Számláló kiírása.

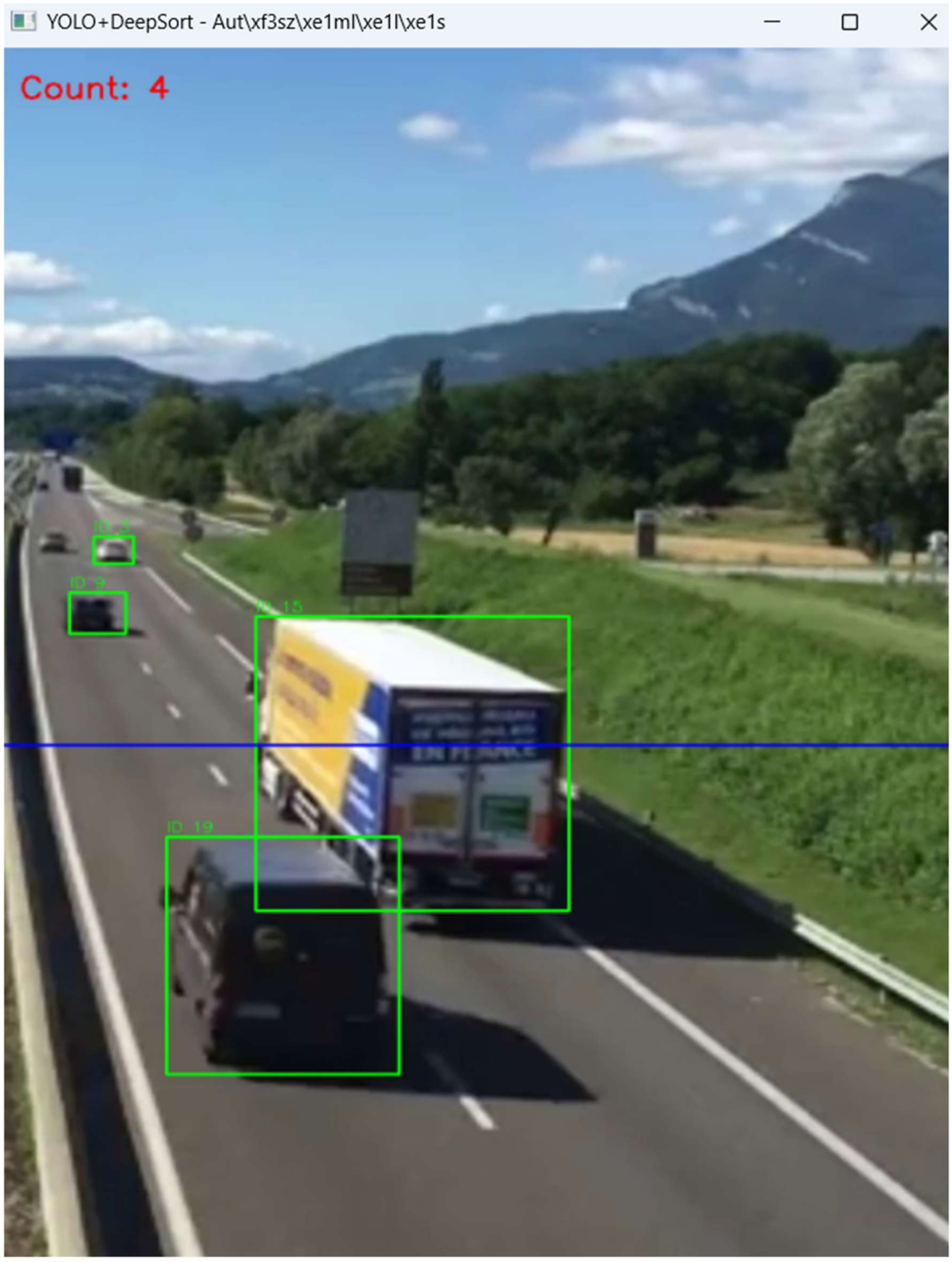
```
    cv2.imshow("YOLO+DeepSort - Autószámlálás",resized)

    # ESC-re kilép
    if cv2.waitKey(1) & 0xFF == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Az ablak megjelenítése/ eltűntetése


YOLO+DeepSort - Aut\xf3sz\xe1ml\xe1l\xe1s
Count: 4
ID: 3
ID: 9
ID: 15
ID: 19



A végeredmény