Videóalapú járműsebesség-becslő rendszer

Video-Based Vehicle Speed Estimation System

Dóka Sándor
WWXN5K

2025

Tartalomjegyzék

1. Bevezetés

A technológia fejlődésével elérhetővé vált számos terület, amelyben használható valamilyen mesterséges intelligencia. Ilyen terület a forgalomirányítás és a forgalom vizsgálata is. Térfigyelő kamerák felvételeiből rengeteg információ szűrhető ki különböző programok segítségével.

Ez a beadandó is ezzel a témakörrel foglalkozik. Egy olyan Python alapú rendszerről van szó, amely gépi tanulás segítségével az elhaladó járművek sebességét becsüli. A rendszer a YOLOv8 mélytanuló objektumdetektáló modellt alkalmazza a járművek felismerésére, a ByteTrack algorimust a detektált objektum lekövetésére. A Supervision könyvtár biztosítja a videókezelés, detekciók és annotációk zökkenőmentes kezelését. Az eredmény egy olyan kimeneti videó, amely járműtípus szerint szín kódolva megjeleníti a jármű sebességét és a fajtáját.

2. Anyag és módszer

2.1. Anyag

A projekt az alábbi szoftveres komponensek és könyvtárak segítségével valósult meg:

1. **Ultralytics YOLOv8:** ez egy nagypontosságú detekciós modell, amely képes járművek felismerésére a videón.
2. **OpenCV:** képfeldolgozáshoz, a video be- és kiolvasására használt könyvtár. Ennek segítségével végzünk perspektívatranszformációt is.
3. **NumPy:** tömbműveletek és egyéb számítások végrehajtására használt könyvtár.
4. **Supervision:** egy a Roboflow csapata által fejlesztett Python-könyvtár, amely egységesít és integrálja a legfontosabb részeit a projektnek (Ultralytics, ByteTrack). Különböző annotációs és zónakezelő eszközök használatát teszi lehetővé.
5. **ByteTrack:** a Supervision könyvtáron keresztül elérhető követőalgoritmus. A detekciókat összekapcsolja egyedi tracker_id azonosítókkal.

A bemeneti adat jelenleg korlátozott egy bizonyos forgalmi videóra, mivel a kamera szöge és egyéb perspektívák torzulása csak ezen a videón lett kiszámolva. A videó adatai, azaz a képkockasebesség, felbontás, videó hossz a Supervision VideoInfo objektumával egyszerűen kiolvasható. Ezek az adatok alapozzák meg a későbbi sebesség számolást.

```
video_info= sv.VideoInfo.from_video_path(args.source_video_path)
```

A sebességmérés egy előre definiált útszakaszra korlátozódik. Ezt a kódban a SOURCE nevű, négy pontból álló koordinátahalmaz definiálja. A célkoordinátát, amely egy magas téglalap alakra hasonlít a TARGET változó reprezentálja. Erre a TARGET által körül írt téglalapra vetítjük rá a SOURCE által leírt poligont.

```
SOURCE= np.array([[1252, 787], [2298, 803], [5039, 2159], [-550, 2159]])

TARGET_WIDTH = 25
TARGET_HEIGHT = 250

TARGET = np.array(
    [
        [0, 0],
        [TARGET_WIDTH - 1, 0],
        [TARGET_WIDTH - 1, TARGET_HEIGHT - 1],
        [0, TARGET_HEIGHT - 1],
    ]
)
```

2.2. Módszer

A program képkockáról képkockára dolgozza fel a videót. A Supervision get_video_frames_generator függvénye segítségével a képkockákat könnyen átadhatjuk a YOLOv8 modellnek, amely feldolgozza őket. Ezt követően a Supervision Detections.from_ultralytics függvénye átalakítja az eredményeket egy egységes, könnyen kezelhető formára. Ez tartalmazza többek között a bounding box koordinátákat és a valószínűségi pontszámokat.

```
frame_generator =sv.get_video_frames_generator(args.source_video_path)

result= model(frame)[0]
detections=sv.Detections.from_ultralytics(result)
```

A SOURCE által kijelölt poligonon belül történik csak sebesség számítás ezzel elkerülve a háttérben is egy pillanatra felbukkanó járművek detektálását. A Supervision PolygonZone osztálya ebben segít, a polygon_zone.trigger(detections) hívás egy maszkot ad vissza, amely segítségével kizárhatók a zónán kívüli detekciók. A fennmaradó detekciókat a ByteTrack követőmodul feldolgozza és tracker_id azonosítóval látja el.

```
detections=detections[polygon_zone.trigger(detections)]
detections= byte_track.update_with_detections(detections=detections)
```

A járművek sebesség becslése a bounding box kitüntetett pontjára épül. A Supervision get_anchors_coordinates metódusa a dobozok alsó középpontját kérdezi le. Ez az egyik legjobban megközelíthető pont, mivel itt „találkozik” a jármű az aszfalttal. Ezeket a pontokat a ViewTransformer osztály használja fel, majd a cv2.getPerspectiveTransform függvény segítségével kiszámítja a SOURCE poligon és a TARGET téglalap közötti mátrixot. A transform_points metódus pedig átszámítja a pontokat a célkoordináta rendszerbe.

```
class ViewTransformer:
    def __init__(self, source: np.ndarray, target: np.ndarray):
        source= source.astype(np.float32)
        target= target.astype(np.float32)
        self.m= cv2.getPerspectiveTransform(source,target)

    def transform_points(self, points: np.ndarray)->np.ndarray:
        reshaped_points = points.reshape(-1,1,2).astype(np.float32)
        transformed_points= cv2.perspectiveTransform(reshaped_points, self.m)
        return transformed_points.reshape(-1,2)

points= detections.get_anchors_coordinates(anchor=sv.Position.BOTTOM_CENTER)
points= view_transformer.transform_points(points=points).astype(int)
```

A sebesség számításhoz minden tracker_id-hoz rendelünk egy deque-t, amely a legutóbbi koordinátát tárolja egy másodpercre visszamenőleg. Ezt a fajta tárolást a deque tudja a leggyorsabban elvégezni, ezért nem használunk tömböt vagy listát. Amíg nem áll rendelkezésre elegendő adat, azaz kevesebb mint fél másodpercig volt csak detektálva addig a sebesség nem jelenik meg ezzel elkerülve a nagyon pontatlan becsléseket.

Miután az objektum több mint fél másodpercig volt érzékelve, a program a deque első és utolsó elemének különbségét tekinti a vizsgált időintervallumban megtett távolságnak. A megtett távolság az adott időintervallumban, mivel a célkoordináta rendszer méter alapú, így méter/másodperc alapú sebességértéket ad. Ezt az érétket 3,6-tal szorozva átváltjuk km/h-ba.

```
coordinates= defaultdict(lambda:deque(maxlen=video_info.fps))

   coordinates[tracker_id].append(y)

   if len(coordinates[tracker_id]) < video_info.fps / 2:
      labels.append(f"#{tracker_id} {class_name}")
   else:
        coordinates_start = coordinates[tracker_id][-1]
        coordinates_end = coordinates[tracker_id][0]
        distance = abs(coordinates_start - coordinates_end)
        time = len(coordinates[tracker_id]) / video_info.fps
        speed = distance / time * 3.6

        labels.append(f"#{tracker_id} {class_name} {int(speed)} km/h")
```

A megjelenítést több a Supervision által biztosított annotátor segíti. A TraceAnnotator a doboz alsó középpontja alapján megjeleníti a megtett útvonalat. A BoxCornerAnnotator a detekciós dobozok sarkait rajzolja ki, a LabelAnnotator pedig kiírja a doboz felett az általunk megjeleníteni kívánt információkat. Ezenkívül az sv.draw_polygon függvény egyértelműen meghatározza a vizsgált zónát a videóban. A Supervision VideoSink komponense a teljesen annotált eredményt kiírja videó kimenetként.

```
corner_annotator = sv.BoxCornerAnnotator(thickness=thickness, color_lookup=sv.ColorLookup.CLASS)
label_annotator= sv.LabelAnnotator(text_scale=text_scale,text_thickness=thickness, color_lookup=sv.ColorLookup.CLASS)
trace_annotator= sv.TraceAnnotator(thickness=thickness, trace_length=video_info.fps * 2,
                                   position=sv.Position.BOTTOM_CENTER, color_lookup=sv.ColorLookup.CLASS)
```

3. Eredmények

Végeredményként a rendszer egy olyan videót hoz létre, amelyen vizuálisan megjelenik a detektált jármű doboza és nyomvonala. A feliratok megjelenítik a ByteTrack által adott azonosítót, A YOLOv8 által meghatározott jármű típust, valamint a sebességértéket. Ez a sebességérték nem jelenik meg rögtön a poligonba való belépés után, így elkerülve a pontatlan sebességbecsléseket adathiány miatt. Természtesen ez a sebesség nagyban függ a SOURCE és TARGET terület pontos meghatározásától.

4. Következtetések és javaslatok

A projekt kitűnően bemutatja, hogy egy jól fejlesztett könyvtár, ebben az esetben a Supervision mennyire tömör és egyszerűen értelmezhető programot eredményez. Ezen kívül a Supervision oldala rengeteg különböző segédanyagot tartalmaz és leírást a könyvtár különböző eszközeiről.

Viszont ez a rendszer jelenlegi formájában sok kihívással küzd. A SOURCE és TARGET meghatározása matematikai háttértudást igényel, valamint nagyon pontos információt a kamera elhelyezkedéséről. A kamera remegése, valamint, az én megfigyelésem alapján a dobozok minimális ugrálása nagyban növeli a becslés pontatlanságát. A SOURCE és TARGET kijelölése továbbfejleszthető egy grafikus felülettel, ezenkívül a becslések stabilizálása lenne a legfontosabb. A program bővíthető lenne további funkciókkal is, mint például forgalomszámlálás, veszélyes baleset közeli helyzetek jelzése.

5. Összefoglalás

A dokumentumban egy Python nyelven megvalósított, videóalapú járműsebesség-becslő rendszer működését és felépítését írja le. A rendszer központi eleme az Ultralytics YOLOv8 detekciós modell, valamint a ByteTrack által biztosított objektum követés. A sebességszámoláshoz jelentősen támaszkodik a Supervision könyvtárra. A rendszer perspektívatranszformációval próbálja megbecsülni a járművek tényleges sebességét. A kimenet egy információban gazdag videó, amely megjeleníti az adott jármű fajtáját szín kódolva, azonosítóját, sebességét, valamint megjeleníti az általuk megtett útvonalat és a mérési zónát.

6. Források

[1] **Roboflow** (2023–2024): Supervision – Open-Source Computer Vision Utilities for Python.

[2] **Ultralytics** (2023): Ultralytics YOLOv8 Documentation.