Autók számlálása kamerafelvételen

Képfeldolgozás haladóknak projekt

# Csapatunk és feladataink:

**Csille Milán Zoltán:** GUI, tesztállomány gyűjtés, tesztelés

**Haag Boglárka Anna:** GUI és számlálás implementálásának átnézése, korrekciója, dokumentáció, tesztelés

**Kolláth István Tibor:** meglévő implementáció finomhangolása, módosítása, további módszerekkel való összevetése, tesztelés

**Sánta Péter:** meglévő implementáció finomhangolása, módosítása, további módszerek keresése, tesztelés

# A feladat specifikációja:

Grafikus felülettel rendelkező program készítése, melyen keresztül a felhasználó kiválaszthat videófelvételeket elemzésre. Az elemzés során a program megszámolja az elhaladó autókat. A program futása alatt a felhasználó végig követheti az elemzési folyamatot, így meggyőződhet az érzékelés helyességéről, majd a felvétel elemzésének befejezése után a program visszajelzést ad a kapott eredményekről.

# Első próbálkozások:

Az objektumok követése, annak detektálása, hogy két képkocka között melyik autó az új, és melyik egy már észlelt autó (csak más pozícióban), korábban máshogy is nézve volt, mint a végső verzióban. Az volt az ötlet, hogy megpróbáltuk összekapcsolni az egyik képkockán található autókat a rákövetkező képkockán találtakkal. Minden kocsi esetén a befoglaló téglalapjának közepét vettük, ezt tekintettük a pozíciójának. Minden képkocka esetén megnéztük az összes pozícióra, hogy tudunk-e társítani hozzá egy pozíciót a rákövetkező képkockáról (egy pozícióhoz csak egyetlen másik pozíció volt társítható), ha igen, akkor úgy vettük, az ugyanaz az autó. Lényegében egy maximális párosítást végeztünk két egymást követő képkocka pozíciói között. Két pozíció akkor volt párosítható, ha Euklideszi távolságuk kisebb volt mint 10/11/12/13 … 99/100 (az összes értékre tesztelve volt a program). Ez a megközelítés sikertelennek bizonyult azonban, bármilyen euklideszi távolságot is vettünk, túl sokszor számolta az algoritmus ugyanazt az autót többször is.

Maguknak az autóknak a detektálására is volt egy másik próbálkozásunk, szabály alapú, különböző előfeldolgozó lépések segítségével: szürkeárnyalatos képpé alakítás, morfológiai eszközökkel ellipszis alakú strukturáló elemek keresésével, dilatációval, árnyékok kivételével, a talált alakzatok méretének figyelembevételével. Azonban a gépi tanuló modell teljesítménye pontosabbnak bizonyult, míg a szabály alapú módszer 45 autó helyett 58-at talált, a legjobb gépi tanulós kísérletünk 44-et talált.

# Végleges program:

## Telepítés:

A programot Python nyelven írtuk és Python 3.9 vagy e feletti verziószámú interpreterrel futtatható. A futtatáshoz a pip install -r requirements.txt parancsot kell futtatni. A szükséges csomagok és verzióik szintén ebben a txt-ben találhatóak.

## GUI:

A grafikus felület elkészítéséhez a PySimpleGUI package-t használtuk. Az ablak teljes képernyős módban nyílik meg és két fő részre bontható. A baloldalon egy, a videófájlokat tartalmazó mappa kiválasztására szolgáló tallózó, valamint a kiválasztott mappa tartalmát megjelenítő ablak van. Az utóbbiban a mappa kiválasztása után megjelent videók közül egyszerű kattintással lehet kijelölni a vizsgálni kívánt fájlt. A videó kiválasztásával a jobboldalon megjelenik a videó első képkockájának képe, valamint egy függőleges csúszka, aminek a mozgatásával beállítható a számlálási sáv magassága. Az algoritmus a képnek csak egy kis részén számolja ténylegesen az autókat, ezt a területet határolja a két kék vonal, így a csúszka beállításánál figyelni kell rá, hogy ezek láthatóak legyenek, valamint az optimális működés érdekében olyan magasságot állítani be, ahol az autók már felismerhető méretűek lesznek, de még nem lógnak le a képről. A megjelenített képkocka alatt látható Count gombbal hívható meg a számolást végző függvény, melynek összesített eredményét a futás befejeztével kiíratjuk a Count gomb alá.

## Számláló algoritmus:

### Előkészítés:

A videó fájl betöltése és a tulajdonságainak lekérése, globális változók létrehozása tartozik ide, illetve létrehozunk egy, a képkockák számának megfelelő sorú DataFrame-et a pandas package segítségével. A képeket a könnyebb megjelenítés és gyorsabb feldolgozás érdekében kicsinyíteni fogjuk. Innentől kezdve pedig egy while ciklusban végig járjuk az egész videót és egyenként elemezzük a képkockákat.

### Detektálás és megjelenítés:

Az aktuális képkocka beolvasása és átméretezése után meghívjuk a   
cvlib detect\_cars\_on\_frame függvényét. Ennek feladata a járművek detektálása.   
A detect\_common\_objects függvénynek átadjuk az adott képkockát, ami a megadott modell segítségével (esetünkben a yolov4 modell vált be a leginkább) a tanult objektumokat detektálja és felcímkézi, és a címke mellett visszaadja a befoglaló téglalapjukat, valamint egy konfidencia értéket. A címkék alapján szelektáljuk a talált objektumokat, csak a car, truck, bus és motorcycle címkével ellátottakat tartjuk meg. A detect\_cars\_on\_frame függvényből három tömbbel térünk vissza, melyeknek sorai a megtartott objektumok fent említett tulajdonságait reprezentálják. A visszakapott befoglaló téglalapokat és a címkéiket OpenCV segítségével megjelenítjük, valamint megjelenítjük a detektálási területet jelző vonalakat is.

### Centroidok meghatározása, objektum követés:

Minden befoglaló téglalap centroidját meghatározzuk az átlóinak metszéspontja alapján, és ezeket a centroidokat használjuk fel az objektum követéséhez. Eltároljuk két tömbben az érzékelési terület (azaz a két kék vonal) közé kerülő centroidokat, és megjelenítjük a képen is ezeknek a helyét. A centroidok követése távolság alapján történik, maximum 25 pixel sugarú körben mozdulhat el két képkocka között úgy, hogy még ugyan annak az objektumnak érzékelje a program. Ha van az aktuális képkockán a vizsgált területen belül centroid, akkor ha még nem volt felvéve detektált autó, azaz a carids tömb üres, akkor felvesszük az összes jelenleg talált objektum sorszámát, valamint az előkészület során létrehozott DataFrame aktuális sorába (megegyezik az aktuális képkocka számával) létrehozunk egy-egy oszlopot a járműazonosítók szerint az objektumoknak, és az oszlopukban eltároljuk a centroid koordinátáit az aktuális képkockán. Ha volt felvéve jármű, akkor megkeressük az összes jelenleg érzékelt centroidhoz, hogy tartozik-e hozzá a df-ben az előző képkockáról egy pár. Ehhez minden centroid esetén végig iterálunk a carids tömbön és megpróbáljuk lekérni a df-ből az előző képkockához tartozó sorból a megfelelő azonosítójú elemet. Ha nincs ilyen, akkor valószínűleg az adott autó elhagyta a megfigyelt területet. Ha volt eltárolva ugyan ezzel a sorszámmal jármű az előző képkockáról, akkor elmentjük az elmozdulását az   
(előző pozíció – jelenlegi pozíció) képlet alapján, mindkét koordináta szerint. Miután minden jelenlegi centroidra meghatároztuk, hogy lehetett-e előző pozíciója is, végigiterálunk csak a carids tömbön, és minden id esetén megnézzük, hogy melyik jelenleg detektált centroidnál a legkisebb az elmozdulás mértéke, majd ellenőrizzük, hogy az adott indexű centroid benne van-e a megengedett elmozdulási tartományban (25 pixeles kör). Amennyiben benne van az adott körben, az adott centroid koordinátáit hozzárendeljük ehhez az id-hoz a df aktuális sorában. Ezután felvesszük őket azok közé a centroidok közé is, amiket hozzá tudtunk adni előző képkockán létező id-hoz. Azokat a centroidokat, amiket nem tudtunk előző képkockán lévő id-hoz kötni felvesszük egy új id-val a df-be.

### Feliratozások, eredmények kijelzése

Ezután a jelenleg érzékelési területen belül lévő járművek centroidjait látjuk el a képen a hozzájuk rendelt id-val, valamint ez után számoljuk azt is, hogy a középső zöld vonalon hány autó haladt át felfelé és lefelé. Az eredményeket a kép bal felső sarkában jelenítjük meg szövegesen is. Az összesen számolt autók számát a carids tömb hosszából tudjuk, ezt a számot a GUI számára visszaadjuk megjelenítésre.

# Eredmények:

A program lassan tölt be a YOLOv4 modell fájljainak betöltése miatt, valamint nem képes CPU-n valós idejű futásra. GPU-n nem volt lehetőségünk tesztelni. Az érzékelés stabilnak mondható, mert az esetek többségében megtalálja a járműveket. Kivételt ez alól olyan esetek képeznek, ahol szokatlan külsejű jármű, például tartálykocsi jelenik meg, ezeket valószínűleg nem tartalmazta a tanító halmaz. A detektált járművek esetén a puttonyos autóknál megfigyelhető, hogy néha car-ként, néha pedig truck-ként címkézi, viszont a számolás során ez nem okoz problémát, mert a centroidok helyzete alapján megfelelően beazonosítja őket és járműnek érzékeli. Az összeszámolt autók száma függ a vonalak pozíciójának kiválasztásától is, ugyanis azokban az esetekben, ahol a kép szélén az autók nem egész méretükben jelennek meg, ott előfordulhat téves érzékelés, és ha ez a terület egybeesik az érzékelési területtel, ott az „újra megtalálások” miatt nőhet a számolt érték. Szintén jelentkezhet probléma utánfutós autóknál, átsorolás esetén, ha túl közel kerülnek egymáshoz az autók, vagy ha a kamera szögéből szinte egybeérőnek tűnnek az autók. Ezekben az esetekben is téves érzékelések jelentkezhetnek, az autók színétől, típusától függően változik, hogy mennyire tudja elkülöníteni őket, például fehér puttonyos autó és fehér személygépkocsi esetén ha túl közel vannak, akkor gyakran váltogatja, hogy egy járműnek (truck) vagy két járműnek (truck+car vagy car+car) érzékeli őket. Az ilyen szélsőséges esetektől eltekintve az algoritmus alapvetően stabil.