TANSZÉKVEZETŐ

DIPLOMATERVEZÉSI FELADAT

Benda Krisztián
mérnökinformatikus hallgató részére

Feliratok illesztése képekre és videókra mesterséges intelligencia használatával

A multimédia és filmes iparban a videók feliratozása, a feliratok elhelyezésének megfelelő kiválasztása fontos feladat, és már régóta igény mutatkozik ennek az automatizálására. Ha a videó tartalmához szeretnénk igazítani a megjelenítendő szöveget, akkor megfelelő feliratpozíció kiválasztása történhet fontos objektumok, például logók, feliratok/szövegek, emberek, illetve a videó egyéb elemeinek és szereplőinek felismerése, behatárolása, relatív helyzetének elemzése, követése révén. A hallgató feladata, hogy ez utóbbinak az automatizálására kidolgozzon és megvalósítson egy olyan mesterséges intelligencia algoritmusokat használó rendszert, mely elkerüli a buktatókat (pl. a felirat ne takarjon ki logót, más feliratot/szöveget, arcot, vagy fontosabb részt a képen) és lehetővé teszi, hogy a kidolgozott elhelyezési startégiát használva, megfelelő időzítéssel játszunk le feliratozott videókat.

A hallgató feladata terjedjen ki az alábbiakra:

- A multimédia feldolgozás szakirodalmán belül mutassa be a feliratozás fontosságát, az automatizálásra használható módszereket és feladat kihívásait!
- Válasszon ki megfelelő képeket és videókat, melyek alkalmasak a feliratelhelyezés változás megfigyelésére. Szükség esetén címkézze fel őket, hogy mesterséges intelligencia használatával is feldolgozhatók legyenek!
- Azonosítson olyan jellemzőket, amelyeket érdemes figyelembe venni a feliratelhelyezés során! Térképezze fel, hogy kiválasztott jellemzők milyen algoritmusok és eszközök segítségével nyerhetők ki a rendelkezésre álló adatfolyamokból!
- Tervezzen meg egy rendszert, mely automatikus felirat elhelyezésre képes a videók képeire! Gyűjtse össze ehhez a különböző algoritmusokat, melyek eltérő szempontok szerint helyezik el a feliratot az egyes képeken a képek tartalmának figyelembevételével, majd adjon saját, mesterséges intelligencia alapú megoldási módszert a meglevő algoritmusok felhasználásával!
- A feliratok elhelyezését két lépcsőben valósítsa meg! Az első lépcsőben csak különálló képeken oldja meg az illesztést úgy, hogy ehhez implementálja a választott algoritmusokat! Második lépcsőben pedig az egymás utáni képeken (azaz a videón) valósítsa meg az illesztést!
- Tárja fel, hogy milyen más szempontok szerint érdemes feliratokat elhelyezni, megbontani (például párbeszéd feliratozása esetén) és adjon megoldási módszert rájuk!
- Definiáljon néhány jósági tényezőt, amik alapján a megalkotott feliratelhelyező rendszer kiértékelhető és végezze is el az értékelést!
- Foglalja össze a munkáját, adjon a rendszer továbbfejlesztésére javaslatokat!

Tanszéki konzulens: Dr. Szűcs Gábor
Külső konzulens: Rechner István, IBM Watson Media

Budapest, 2019. szeptember 30.

/ Dr. Magyar Gábor /
tanszékvezető

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék
1117 Budapest, Magyar tudósok krt. 2.
Telefon: +36 1 463 2448 Fax: +36 1 463 3107
E-mail: magyar@tmit.bme.hu http://www.tmit.bme.hu

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és médiainformatikai tanszék

Benda Krisztián

FELIRATOK ILLESZTÉSE KÉPEKRE ÉS VIDEÓKRA MESTERSÉGES INTELLIGENCIA HASZNÁLATÁVAL

KONZULENS

Dr. Szűcs Gábor

BUDAPEST, 2020

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Benda Krisztián**, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2020. 05. 16.

...…….

Benda Krisztián

Összefoglaló

Az egyre növekvő igény az internetes tartalomfogyasztás iránt a videó készítési és feldolgozási folyamatokat is felgyorsítja. Videók mellett egyre nagyobb hangsúlyt kap a feliratok generálása és használata is. Manapság már nem csak a süketek, nagyothallók és idegennyelvűek számára készülnek szövegezési megoldások, hanem az átlagembernek is, hogy különböző helyzetekben például zajos környezetben könnyítse meg a tartalomfogyasztást.

A feliratok pozíciója számos esetben zavaró helyre esik, amely gátolja az alatta lévő képi információ átadását. Sokszor letakar más a képen megjelenő szöveget, vagy fontos vizuális tartalmat. Dolgozatomban bemutatok egy okos feliratpozícionáló megoldást, melynek használatával elkerülhetjük a zavaró feliratpozíciókat képeken és videókon egyaránt.

Először a témakört és a videó feliratozási problematikát ismertetem, majd a kapcsolódó tudományos cikkeket elemzem részletesen. Detektálható képtulajdonságok alapján rendszerarchitektúrát terveztem, amely képes elvégezni az okos pozícionálást képeken és videókon. A képfeldolgozó alrendszert felhasználva elkészítettem a videó feliratpozícionáló megoldást is. Különböző algoritmusokat mutatok be, amelyek több szempont alapján képesek a felirat elhelyezésére videókon. A megoldásomhoz jósági tényezőket definiáltam és egy közvéleménykutatás segítségével értékeltem ki a különböző futtatási lehetőségeket. Végül az eredmény bemutatása után kitekintést nyújtok a hasonló témakörökről és továbbfejlesztési javaslatokat teszek, hogy munkám széleskörben is felhasználható legyen.

Abstract

The growing need for Internet content consumption also accelerates video production and processing. Besides the increasing number of videos and films, generating and using captions are also got emphasis. Nowadays, not only deaf, hard of hearing, and people speaking other foreign languages targeted by captioning solutions. The average people are also aimed because facilitating content consumptions is required as well with better transcription solutions in various situations such as noisy environments.

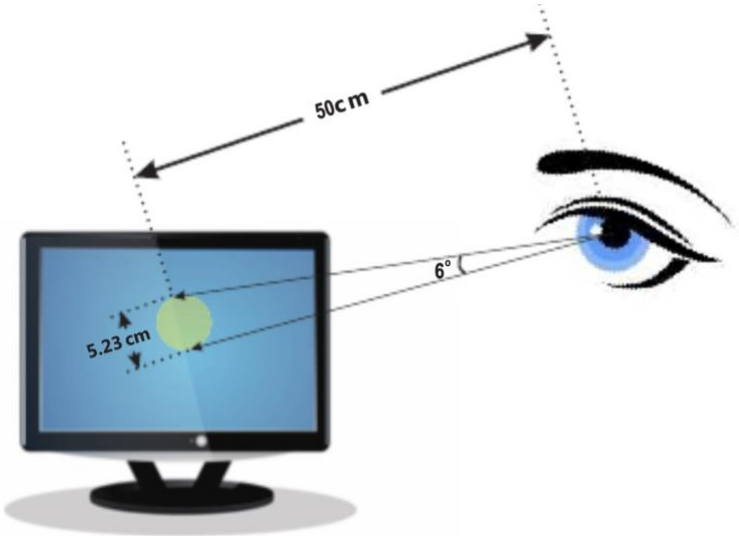
In many cases, the position of the subtitles and captions are in a confusing area, which prevents the transmission of the underlying visual information. Many times, it obscures other text or important visual content in the image. In my thesis work, I present a smart subtitle positioning solution that avoids annoying subtitle positions on both images and videos.

First, the topic and the problem of video captioning presented, then the related scientific articles analyzed in detail. Based on detectable image properties, I designed a system architecture that can perform smart positioning on images and videos as well. Using the image processing subsystem, a video caption positioning solution was implemented. Different algorithms described in detail that are able to place captions on videos based on several aspects with the designed system. For the solution, various goodness factors were created. With the help of the factors and a specific poll the evaluation of the system's running options was made. Finally, after presenting the results, an overview of similar topics is provided and further suggestions for the development highlighted ensuring that my work can be widely used.

1 Bevezetés

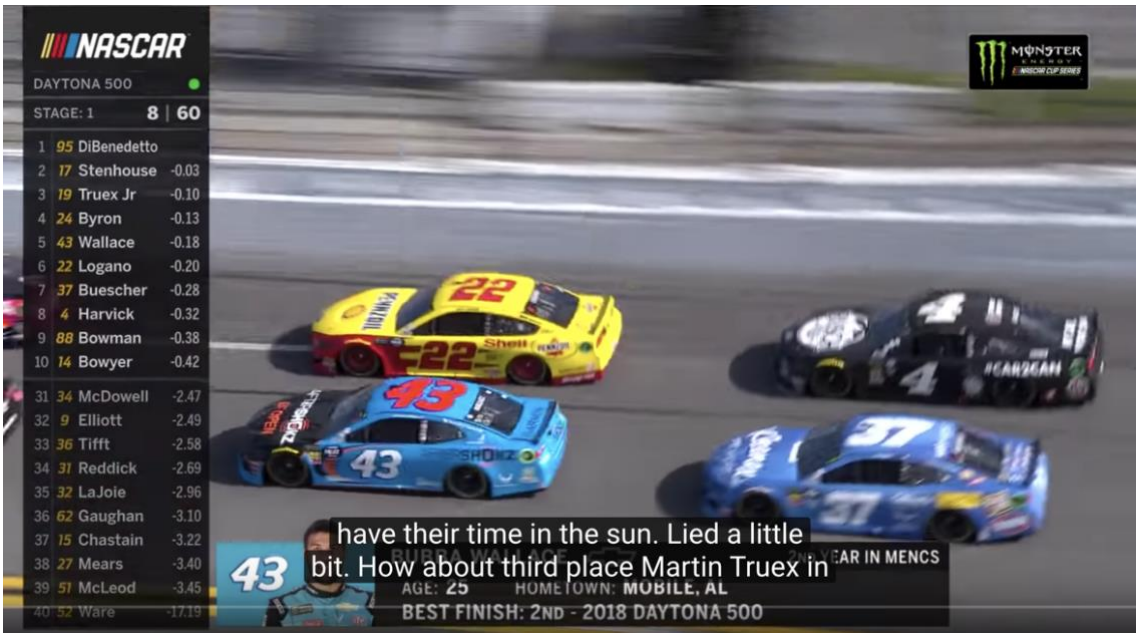
Napjainkban a multimédia tartalmak egyre szélesedő körben használatosak. Az emberek már nem csak televízió, számítógép előtt néznek filmeket, videókat, hanem sokszor tömegközlekedve kisebb-nagyobb képernyőkön is fogyasztják a vizuális tartalmat. Ahogy nő az igény a videók iránt, úgy kap egyre nagyobb szerepet a videókhoz tartozó felirat is. Feliratok videóra helyezése különböző célokat szolgálhat. Idegen nyelvű videóknál, hallássérülteknél, audió lejátszására nem alkalmas, zajos környezetben (például tömegközlekedés közben, utcai kivetítők esetén) tagadhatatlan a felirat szükségszerűsége.

Legtöbb esetben a videóhoz tartozó felirat egy fix pozícióba van égetve a képen. Általában ez a videó alján, középre igazítva jelenik meg, amely sokszor optimális megoldás lehet, de sok „felesleges” szemmozgásra kényszeríti a nézőt. Az emberi szem fókuszpontja, amely az olvasást teszi lehetővé, nagyon keskeny. Egy fókuszpont használatával egy-egy szót tudunk csak értelmezni (ez természetesen emberenként eltérő lehet). Fél méterről 5,23 cm átmérőjű körnek felel meg (1.1 ábra). Ezért a néző mozgalmas, sok párbeszédet tartalmazó részeknél gyakorlatilag nem tudja szemmel követni a cselekményt, ha a feliratot követi tekintetével, ami rontja a tartalom élvezhetőségét és érthetőségét is. Továbbá a gyakori fókuszváltás a felirat és a vizuális cselekmény között, a szem megerőltetésével jár együtt.



1.1 ábra: Az emberi szem korlátozott látótávolsága. Fél méterről 5,23 cm átmérőjű kör [1]

A felesleges szemmozgásokon kívül más aspektusban is fontos a felirat pozíciója. Vannak olyan tartalmak, például híradó, sportközvetítések mikor a beszélő nem is szerepel a képen, csak a lényegi tartalom látszik, de számos más információ van közölve a képernyő szélén. Erre mutat példát a következő ábra (1.2 ábra ábra). Jól látható, hogy az autóversenyzés közben fontos adatok jelennek meg a képernyőn az egyes versenyzőkről és az autók aktuális sorrendjéről. A háttérben a kommentátor folyamatos plusz információval gazdagítja a versenyt ezért, ha bekapcsoljuk a feliratot a videóhoz, gyakorlatilag elveszítjük a képernyőn megjelenő információk egy jelentős részét. Egyértelműen látszik, hogy a kép felső része alkalmasabb lenne a felirat elhelyezésére, mert ott csak az elmosódott környezet látszódik.



1.2 ábra: Autóversenyzés a kommentátor szövegének feliratával

Már csak a fentebb említett két probléma kapcsán is érdemes elgondolkodnunk a felirat pozíciójának optimalizálásán, dinamikus igazításán. Mivel figyelmemet felkeltette az ismertetett problematika, ezért diplomamunkámmal mélyebben is elkezdtem foglalkozni a feladattal és megoldási lehetőségeivel.

1.1 A témakör ismertetése

A videó és film feliratozás problémakör nagyon sokrétű, melyet több részfeladatra tudunk bontani. A legfontosabb lényegi elem egyértelműen a ***felirat szövegi tartalma***. Ezután jön a ***szövegezés pontos időzítése*** és igazítása a hangsávhoz, majd ***pozícionálás*** a videón megfelelő vizuális megjelenéssel. Ezek közül diplomamunkám a megfelelő

pozícionálással foglalkozik széleskörűbben, de különböző részletességgel a másik két feladatkörre is kitérek. Viszont a részfeladatok előtt érdemes megismerni a különböző felirattípusokat.

1.1.1 Felirattípusok

A magyar nyelv kevésbé tesz különbséget az angol *caption* és *subtitle* szavak jelentése között, hisz mindkettőre a felirat szót használjuk, holott a két szó jelentése eltérő idegen nyelven [2]. A *caption* szó általában süket és nagyothallók számára készített, a videó nyelvével azonos feliratot takar. A *subtitle* pedig a feliratba ágyazott fordítást teszi lehetővé. Tehát a *subtitle* elsősorban nem hallássérülteknek készül, ezért sokszor nem találhatok meg benne hangutánzó szavak, nem emberi hangok feltüntetései. A *caption* pedig a videó eredeti nyelvén érhető el és hangleíró szavakat és kifejezéseket minőségtől függően tartalmazza. Mivel én elsősorban a felirat elhelyezésével foglalkozom, ezért az én szempontomból neutrális, hogy *caption*-nel, vagy *subtitle*-lel kell dolgoznom, ezért a két kifejezést dolgozatomban én is szinonimaként használom (akárcsak a magyar nyelv).

Az angol szakirodalom megkülönböztet két másik felirat típust is. Az *open* és a *closed captioning*-et. *Open captioning (OC)* esetén a felirat kikapcsolására nincs lehetőség, az a videóba égetve jelenik meg a kép részeként. A *closed captioning (CC)* pedig opcionális és a néző által aktivált feliratot takar [2] [3]. Mind a két típusnak vannak előnyei és hátrányai, melyek munkám szempontjából is igen fontosak, ezért ezeket egy táblázatban foglaltam össze (1. táblázat).

	Open Captioning	**Closed Captioning**
Használat, aktiválás	A hozzá nem értő felhasználónak nem kell külön bekapcsolnia, de kikapcsolni sem lehet.	Ki-be kapcsolható, de minimális hozzáértés szükséges hozzá, mely televíziónként és lejátszónként változhat.
Időzítés	Mindig együtt mozog a hanggal, ezért nem fordul elő aszinkronitás.	Gyakran elcsúszik a videó hangjától, de a sebesség lejátszás közben is állítható.
Pozíció	Készítésnél változtatható, ezért beállítható úgy, hogy ne takarjon ki lényeges tartalmat.	Alaphelyzetben (középen, alul) zavaró lehet. Pozíciója fájlformátumok támogatottságától függ.
Tárolás	Nem kell külön tárolni, mozgatni.	Külön tárolás és fájlformátum szükséges.
Cserélhetőség	Nem cserélhető más nyelvre, de mellette alkalmazható close caption. Viszont ez nagyban rontja a felhasználói élményt.	Könnyen leváltható egy másik nyelvű verzióra.
Fájlformátum, dekódolás	Nem szükséges külön dekóder hozzá.	Különböző fájlformátumok: *.srt, .scc, .ttml, .dxfp, .vtt, .cap, .ass*, melyekhez különböző dekóder szükséges és a lejátszók különbözőket támogatnak.

1. táblázat: Open és Closed Captioning összehasonlítása

Ahogy látható (nevével ellentmondásos módon) a *Closed Captioning* egy sokkal kötetlenebb feliratozási módot tesz lehetővé, ahol a szöveg szabadon módosítható, ki-be kapcsolható. Az én szempontomból hátrány, hogy a legnépszerűbb feliratfájlformátumok nem támogatják a feliratszegmens pozícionálását. Egy feliratszegmensnek azt a felirat szövegrészt nevezzük, amely egy megadott intervallumon belül egyszerre jelenik meg a videón. Ez lehet egy mondat, vagy rövidebb kérdés-válasz, de akár egy hosszabb félmondat is.

A médialejátszókban általában csak egy konstans pozíciót lehet megadni a felirat helyének, mely nem variálható. Tehát ha én feliratszegmens specifikusan szeretném változtatni a pozíciót, akkor vagy egy egyedi fájlformátumot kell alkalmaznom (*ASS: Advanced Sub Station Alpha*), vagy *Open Captioning* feliratozási technikát használnom. Az *ASS* formátumnak kisebb a támogatottsága, ezért szerettem volna az *Open Captioning* megoldást előnyben részesíteni.

1.1.2 Feliratozás folyamata

A ***felirat szövegi tartalma*** különböző forrásból származhat. Napjainkban is gyakori, hogy a feliratozást dedikált ember végzi egy-egy élő videó közvetítése közben, vagy egy önkéntes jelölt fordít le filmeket feliratozott formában. Szerencsére már léteznek hatékony megoldások az automatikus generálására is. Ilyenkor gyakorlatilag egy *speech-to-text* (élő beszéd szöveggé konvertálása) problematikáról van szó, melyeket mesterséges intelligencia algoritmusokkal már egész jól meg tudunk oldani. Rendelkezik ilyen szolgáltatással a Google [4], az Amazon [5], az IBM Watson [6], de ha szeretnénk találhatunk nyílt forráskodú lehetőségeket is [7].

A ***felirat időzítés*** problematikára legkézenfekvőbb megoldás lenne az *Open Captioning* használata, de ez jellegéből adódóan sok kompromisszumot rejt magában. *Closed Captioning* esetén a feliratfájlformátumok, a médialejátszók sokfélesége, illetve a különböző videó kiterjesztések miatt a vetítés közbeni szöveg teljesen aszinkron módon tud mozogni a hanggal. Egyes lejátszók (pl.: VLC [8]) képesek a lejátszás közben is időben tolni a feliratozáson, amely nagy könnyebbséget jelent, de sokszor nem oldja meg teljesen a problémát. Találhatunk olyan módszereket is, amelyek a mesterséges intelligencia előnyeit használják ki és egy adathalmaz alapján hálózatot tanítanak fel a problematikára [9].

A harmadik megoldandó feladatkör pedig a ***pozícionálás***, amit fő csapásiránynak jelöltem ki diplomamunkám során, ahogy már korábban is írtam. Legelterjedtebb megoldásként a videó alján, középen szokták megjeleníteni a feliratot, de ez sokszor fontos információveszteséggel jár. Ahogy az ismertetett ábrán is látható (1.2 ábra ábra). Megoldásként egy olyan algoritmust dolgozok ki, amely képes képeket és videókat elemezni tartalmuk alapján és a feliratot olyan helyre pozícionálja, ahol semmilyen lényeges információt, logót, szöveget, objektumot nem takar ki, amennyiben ez lehetséges.

1.2 A feladat részletes bemutatása

Munkámmal szeretnék egy olyan megoldást tervezni és megvalósítani feliratok elhelyezésére képeken és videókon, mely az eddigiektől eltérő módon, automatikusan elemezi a vizuális tartalmat emberi interakció nélkül és több szempont alapján megvizsgálva pozícionálja a szöveget. Jelen fejezetben a diplomatervezési feladatomat taglalom részletesebben és ismertetem végrehajtásának lépéseit.

A témakör ismertetésénél (1.1 fejezet) már bemutattam a feliratozás fontosságát, körülményeit és a különböző lehetőségeket. A továbbiakban először az irodalomkutatási rész olvasható, melyet a feliratpozícionálás témakörében végeztem. Megkerestem a releváns cikkeket, kutatásokat és elemzem őket a diplomamunkámra vonatkozóan. Itt kitérek a létező automatizálási módszerekre és optimalizálásra szoruló részletekre. Az ismertetett források tartalmaznak olyan pozícionáló eljárásokat, amelyek a párbeszédek könnyebb értelmezése céljából születtek meg és az aktuális beszélőhöz mérten helyezik el a feliratot.

A munka elvégzéséhez megfelelő vizuális tartalmat keresek, melyeknél egyértelműen megfigyelhető a felirat zavaró elhelyezése (egy ilyen már az első példa ábrán [1.2 ábra] is látható). A dolgozatomban külön-külön, a releváns részeknél több ilyen példa is megtalálható, de a külön fejezetet, hosszabb bemutatást nem látom indokoltnak.

A problémakört részletesebben megismerve elmondható, hogy tanító adathalmaz készítése képek felcímkézésével és egy alkalmas hálózat építése kifejezetten pozícionálási célokra optimalizáltan nem lenne célszerű. A feliratok legjobb pozíciójának helyzete szubjektív és nagyban függ a vizuális tartalomtól, ezért nagyon sok manuális munkára lenne szükség és az eredményben ez nem tudna kellőképpen tükröződni, hisz mesterséges intelligencia alapokon más megközelítéssel precízebb és determinisztikusabb megoldást lehet készíteni. Ettől függetlenül címkézett képeket és videókat használtam a munkám során, de azok nem a pozíció célváltozót tartalmazták közvetlenül, hanem más jellegű (pl. objektum típus) címkéket, amit közvetett módon kívántam felhasználni. Ennek az megvalósítását és háttérmunkálatait szeretném bemutatni diplomamunkámban.

Ahhoz, hogy egy dinamikusan változó videón feliratokat tudjunk pozícionálni, a vetített képeket kell megvizsgálni. Ezért a munkámban határozottan elkülönítettem a lejátszás közben megjelenő képekkel foglalkozó alrendszert és a videófeldolgozást

megvalósító modult. Külön tervezési, megvalósítási és értékelési fejezetekben számolok be a kép és videófeldolgozás részleteiről.

A tervezés taglalása előtt azonban rövid irodalmi áttekintést adok a feliratpozícionálás témakörében. Ezek utána a rendszerterv első szakaszában azonosítok olyan jellemzőket, képtulajdonságokat, melyeket érdemes figyelembe venni képeken történő feliratpozícionálásnál. A tulajdonságok automatikus felderítéséhez olyan algoritmusokat párosítok, melyekkel könnyen kinyerhetők tulajdonság információk a képekből. A képi attribútumok kiemelése mellett megtervezem a később megvalósításra kerülő feliratpozícionáló rendszert először képekre úgy, hogy az hasznosítható legyen videós környezetben is. A használhatóságát a tervezés második szakaszában ismertetem a videófeldolgozó modul keretében, ahol a feliratugrálás problematikájáról és megoldási lehetőségeiről is beszámolok részletesen. A megvalósítási rész első fejezetében mesterséges intelligencia alapú algoritmusokat használok, amelyek lehetővé teszik a modern képelemzést, illetve egyéb képfeldolgozási eljárásokat is figyelembe veszek. Itt fontos megjegyezni, hogy az általam használt intelligens algoritmusok nem a felirat pozíciójának célváltozóként való felhasználását célozták meg, hanem alacsonyabb szintű (objektumok, képi információk, stb) tanulási képességét oldották meg. A második fejezetben pedig a videó feliratpozícionálás megvalósításába adok betekintést. A megtervezett és implementált rendszerhez jósági tényezőket azonosítok és ezek alapján plusz egy rövid közvéleménykutatás segítségével kiértékelem a megoldás működését és kiemelem használati lehetőségei közül a legjobb opciókat. Végül kitekintéssel, tovább fejlesztési javaslatokkal és összefoglalóval zárom a munkámat.

2 Irodalomkutatás feliratelhelyezéshez

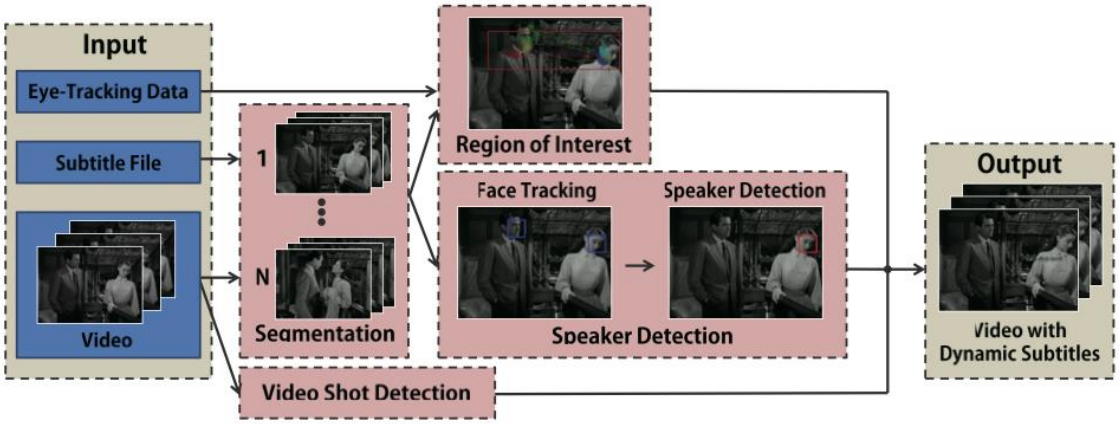
Videó feliratozás igen nagy témakörnek tekinthető a már ismertetett részfeladatok alapján (1.1.2 fejezet). A három feladat közül a Speech-To-Text problematika örvend a legnagyobb népszerűségnek, melyre már nagyon sok jól használható megoldás létezik. A feliratok automatikus pozícionálása azonban bőven nem rendelkezik ekkora szakirodalommal. Lokalizációra legelterjedtebb megoldás a különböző szoftverek által biztosított manuális beállításra/igazításra alkalmas környezet [10].

Az elérhető források a feliratpozícionálást elsősorban párbeszédközpontúan közelítik meg. Olyan megoldásokat kínálnak, amely a párbeszéd könnyebb követését tesszik lehetővé. Intelligens pozícionálásra, amely kifejezetten a felirat által kitakarandó területet analizálja és az optimumot keresi, nem találtam példát.

Japán egyetemek tudósai a videó feliratelhelyezés problematikáját különleges megközelítésben vizsgálták. A felirat helyzetének meghatározásához, nem csak a videón megjelenő vizuális elemeket használták, hanem a nézők fókuszpontjait is figyelembe vették. A kutatás célja a felesleges szemmozgások minimalizálása volt, melynek eredményét és folyamatát egy tudományos cikkben publikálták [11].

A videón megjelenő képeken úgynevezett region of interest (ROI) halmazokat definiáltak, melyek a kép érdekes, fontosabb információit tartalmazzák egy téglalap alakú területbe zárva. A kutatásban kidolgozott eljárás a ROI terület alatt helyezi el közvetlenül a feliratot. Így a néző számára az érdekesebb részek nem kerülnek kitakarásra. Továbbá az algoritmus hang és vizuális elemek alapján az aktuális beszélő személyt is behatárolja a képen és a feliratot a beszélőhöz közelebb helyezi el.

Tehát a megvalósított eljárás inputként egy videót, egy felirat fájlt (*SRT-t*) és egy az előre rögzített szemmozgásokat leíró fájlt kap. A kimenet pedig egy videó, amin megtalálható a dinamikusan változó felirat. Az eljárás sematikus felépítése látható az alábbi ábrán (2.1 ábra ábra). A videó szegmensekre bontása mellett, a jelenetek detektálása is megtörténik Apostolidis és Mezaris technikájával [12]. Ezek után a figyelmi területek (ROI) kiszámításra kerülnek és az aktív beszélő azonosítása is lezajlik. Végül a ROI és a beszélő helyzetének felhasználásával a felirat beégetésre kerül a videóba.



2.1 ábra: Sematikus ábra a szemmozgás alapján történő feliratpozícionálásnak [11]

A ROI kiszámítása az én munkám szempontjából is fontos kérdés. Emberi szemmozgások felvétele túlságosan hosszú és költséges feladat, illetve fenntarthatósági szempontból sem előnyös megközelítés. A tanulmányban ismertetett módszer előnye a pontos statisztikai adat, amely ténylegesen a néző szempontjából fontosnak tartott figyelmi területeket írja le az egyes képkockákon (ROI). A gépi algoritmusok ennél sokkal irányítottabban, például objektumdetektálással tudnak csak dolgozni.

6 angol nyelvű 2 perces videót választottak ki tesztanyagként, melyek 2 film ("*Roman Holiday*" és "*Charade*") néhány jelenetét tartalmazták. A tesztalanyok japán emberek voltak alapvető angoltudással. A vizsgálat alatt a tesztalanyok randomizált sorrendben nézték végig a videókat, miközben szemmozgásukat folyamatos megfigyelés alá vetették.

A tanulmányban feliratszegmensekként történt meg a ROI kiszámítása, így a szöveg szegmensek ugyanabban a pozícióban maradtak megjelenésük alatt, csak újabb feliratsor esetén történt pozíció változás. Más tanulmányokban volt arra is példa, hogy a mindenkori aktív beszélőhöz igazították a feliratot, ami azt eredményezte, hogy a feliratszegmensek együtt mozogtak a videón megjelenített történésekkel [13]. Ez a módszer kevésbé bizonyult effektívnek a túl sok szövegmozgásnak köszönhetően, ezért maradtak a feliratszegmens alapú ROI számításnál, de a későbbi kiértékelésnél ezzel a technikával is összehasonlították a megalkotott eljárást.

Az összegyűjtött szemmozgásokat aggregálták és egy téglalap alapú területtel kerítették körbe. Ezt tekintették a figyelmi területnek (ROI). A fókuszpontok 95%-a esett a területbe, így egy szűkebb intervallumot tudtak meghatározni, az átlagtól távolabb eső fókuszpontok kizárásával.

A pontosság növelése érdekében a tanulmányban felhasználtak Speaker Detection (aktív beszélő meghatározás) és Face Detection (arc detektálás) algoritmusokat is, melyeket később figyelembe vettek a felirat pozícionálásánál.

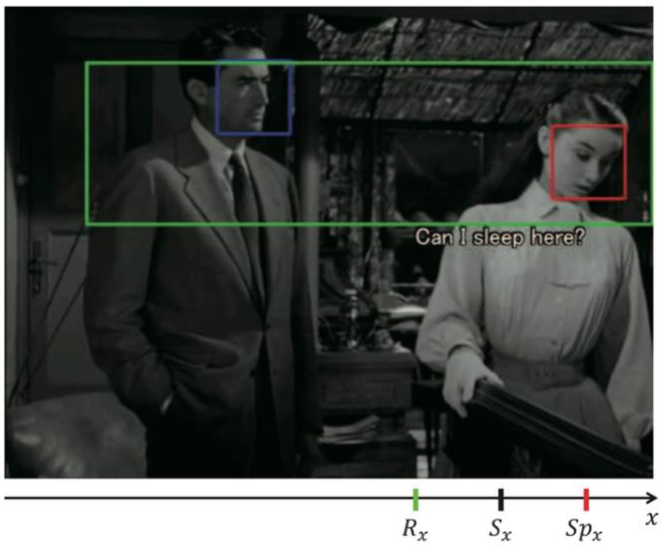
A szöveg elhelyezésnél az alábbi három szabály alapján dolgoztak fontossági sorrendben.

1. A felirat pozíciója alapján a beszélő könnyebben behatárolható legyen, azaz minél közelebb essen a hangforrásához.
2. A felirat ne takarja ki a lemért érdekesebb részeket, ROI területeket.
3. Az egymásutáni feliratszegmensek pozíciója minimálisan változzon.

A feliratot minden esetben a ROI terület alá helyezték, ezzel elkerülve a szöveg jobb, illetve bal szélre (ki)szorulását. Mivel rendelkezésre álltak a ROI és a Face Detection-ből származó adatok, ezért a felirat közepének x-koordinátáját (S_x) az alábbi képlet alapján határozták meg.:

$$S_x \begin{cases} \frac{(Sp_x + R_x)}{2}, & \text{ha az aktív beszélő meghatározható} \\ R_x, & \text{ha az aktív beszélő } \mathbf{nem} \text{ meghatározható} \end{cases}$$

Tehát amennyiben a beszélő arc helyzete is ismert, akkor a felirat az arc (Sp_x) és a ROI (R_x) középpontja között helyezkedik el félúton. Ellenkező esetben a ROI közepénél. Erre mutat példát az alábbi 2.2 ábra:



2.2 ábra: Felirat a ROI középpont (zöld téglalap középpontja) és a beszélő arca (piros) között [11]

Amennyiben rövid idő (0,3 másodperc) telt el két felirat szegmens változása között, a felirat y koordinátája változatlan maradt, hogy a nézőt ne zavarja meg a túl sok pozíció változás. Ha jelenetváltás is tartozott egy darab szövegszegmenshez, akkor a megszokott alul-közép pozícióban maradt.

Az eredmények kiértékeléséhez 19 önkéntest kértek fel, alapvető angoltudással. A videókat 3 féle feliratozási eljárással is lejátszották nekik. Tradicionális, statikus feliratozással, beszélőt közvetlenül követő feliratozással és a tanulmányban kidolgozott összetettebb technikával. Az eredmények kimutatták, hogy szignifikánsan sikerült csökkenteni a szemfixációs időt az új eljárással. Illetve a tesztalanyok 6 videóból 4 esetében egyértelműen kényelmesebbnek és használhatóbbnak ítélték meg a kidolgozott eljárást, míg a maradék két esetben nem éreztek nagyobb javulást az eddig használt technikákhoz képest.

A forrást [11] nagyon fontosnak tartom az én munkám szempontjából, ugyanis a tanulmány íróinak sikerült olyan új feliratozási eljárást kidolgozni, ami emberi mérések alapján javította a felhasználói élményt amellett, hogy a videók lényeges elemei sem kerültek kitakarásra. A technika igen nagy hátránya, hogy használatához először fel kell mérni a nézők fókuszpontjait az egyes képkockákon, ami gyakorlatilag használhatatlanná teszi a tömeges alkalmazást. Saját munkámmal megoldást szeretnék nyújtani erre a problémára. Célom, hogy ne kelljen emberi erőforrást "pazarolni" arra, hogy a képkockák lényegesebb részeit meghatározzuk.

Érdekes, hogy három évvel előbb, 2014-ben kiadott cikk, a *Speaker-following Video Subtitles* ugyanúgy a beszélő közelében elhelyezett felirat problematikát taglalja [1] azonban ekkor még nem foglalkoztak a ROI meghatározásával, így hogyha egy beszélő közvetlen közelében más fontos tartalom van, azt könnyen kitakarja a felirat. A tanulmány nagy hangsúlyt helyez a *Speaker Detection*-re, amely az én szempontomból is érdekes, bár munkám során nem a beszélő pontos beazonosítása a cél.

A kutatás szerint a legelterjedtebb módszer beszélő meghatározására a szájmozgást azonosító algoritmusok, de ezek sokszor nem elég pontosak. Ennek a javítására dolgoztak ki két módszert: a *center contribution*-t és *length consistency*-t. A *center contribution* esetén azzal a feltételezéssel számolnak, hogy az aktuális beszélő valószínűleg közelebb helyezkedik el a kép középpontjához, ezért kérdéses esetekben a középponthoz közelebb eső személy az aktív beszélő fél. *Length consistency*-nél pedig azt vizsgálják, hogy az aktívnak jelölt személy mennyi ideig szerepel a képen a

kapcsolódó felirat szegmenshez képest. Minél nagyobb arányban található meg a képen annál valószínűbb, hogy tőle származik a hang. Ezenkívül *Audio-Visual Synchrony* [14] eljárást használtak a pontosabb eredmények eléréséhez, amely azon alapszik, hogy a képi és hangi mozgás ritmikája összehasonlítható és egy szinkronizációs számmal leírható. A detekciós algoritmusokat sorba kötötték és akkor fogadtak el valakit aktív beszélőnek, ha mindegyik algoritmusnak megfelelt.

A forrás a videókon történő feliratpozícionáláshoz adott számomra ötletet. A megoldásuknál a felirat szorosan az aktív beszélőhöz közel helyezkedett el, ami sok zavaró esethez vezetett. Például mikor a beszélő a kép egyik oldaláról a másikra mozog, a hozzá közeli szegmens kicsúszhat a képből, illetve a néző számára is zavaró folyamatosan mozgó feliratot olvasni. Így inkább rövidebb részekre bontották a szegmenset és több részletben jelenítették meg.

A gyors jelentváltásoknál előfordulhat, hogy a szegmensek egymástól elég távol helyezkednének el egymás után, ami zavaró szemmozgást eredményez. Ezt pedig úgy oldották meg, hogy az új feliratot inkább az előző helyére tették, mintsem egy távolabbi helyre. Illetve azt a problémát is kezelték, mikor a beszélő a jelenetváltásoknak köszönhetően kikerül a képből. Ilyenkor inkább alaphelyzetbe (alul, középen) hagyták a szegmenset.

Anélkül, hogy részletekbe merülnénk elmondható, hogy a 219 tesztalany, akikkel értékelték a megoldást, egyértelmű javulást érzékelt a korábbi feliratozási eljárásokhoz képest. A cikkben leírt módszer a beszélő helyzetére optimalizál, ezért nagyon hasznos olyan környezetben, mikor sok párbeszéd található a videóban, de egy sportadásnál (1.2 ábra) ez inkább zavaró lenne, mintsem hasznos.

3 Tervezés

A munkálatokat két főbb részre osztottam. Először képeken történő feliratelhelyezésre, utána a videókra optimalizált megoldásra. A két feladat problematikájában is különbözik. Képeknél elsődlegesen az a feladat, hogy felismerjük a megjelenő fontosabb elemeket és megfelelő pozíciót keressünk a feliratszegmensnek. Videóknál pedig az összhangra is kell figyelni. Ahogy az egymás utáni képek jelennek meg a videón a felirat hajlamos lenne az „ugrálásra". A különböző jelenetek és kameraállások miatt mindig más a legjobb pozíció. Tehát ennél a résznél arra kell figyelni, hogy a különböző szegmenspozíciók ne okozzanak kellemetlenséget a felhasználó számára. Ezek alapján a két fő feladat a következő:

1. Képeken feliratszegmens okos elhelyezése - képfeldolgozás
2. Videókon feliratszegmens pozíció optimalizálása - videófeldolgozás

A két különálló feladatot a jelen (*3 Tervezés)* fejezetben is külön részre bontottam, azzal a kiegészítéssel, hogy a videófeliratozó rendszer almodulként használja a képfeliratozó megoldást is: 3.1 és 3.2 fejezet.

3.1 Képfeliratpozícionáló rendszer tervezése

Képek információhordozó ereje sokkal több az ember számára, mint amit az egyszerű számítógépek pixelek tárolásával kezelni tudnak. Ahhoz, hogy egy-egy kép vizuális tartalmát jobban megértsék a számítógépek, különböző detekciós eljárásokra van szükség. A detektálható tulajdonságok és feldolgozási lehetőségeik igen fontosak a megvalósítandó rendszer számára, ezért elsőként ezeket ismertetem.

3.1.1 Képtulajdonságok

Ha emberi szemmel ránézünk egy képre általában könnyen meg tudjuk határozni, hogy mi a kép legfontosabb, legérdekesebb része még akkor is, ha az nem éppen a középpontban helyezkedik el. Olyan helyzetek is előfordulnak, amikor detektálni tudjuk, hogy egy területet lefedésével súlyos tartalmi információt veszítünk, annak ellenére, hogy nem a főbb objektum a képen. Ilyenek például a híradós adások, amikor egy konstans szövegdobozban a legfrissebb híreket tudjuk elolvasni, de az aktuális vizuális tartalom másik hírről számol be éppen (3.1 ábra). Sokszor a márkajelzésekkel, logókkal is ez a

helyzet. Kitakarásuk nem szolgálja a jelölt cég érdekeit. Ebben a fejezetben összegyűjtöttem a legfontosabb képi tulajdonságokat, amiket figyelembe kell venni felirat elhelyezésénél, a detektálásukat segítő megoldásokkal együtt.

3.1 ábra: Híradós adás, ahol a felirat alapértelmezett pozíciója már meglévő szövegi tartalomra esik

Képbe égetett szöveg. A fenti képen (3.1 ábra) jól látszódik, hogy a legrosszabb megoldás lenne a feliratot alapértelmezett pozíciójában hagyni. Nem csak az eredeti szöveget nem lehetne elolvasni, de a felirat értelmezésénél is igen zavaró hátteret eredményezne. Tehát a képeken/videókon található szöveget mindenképpen figyelembe kell venni a felirat elhelyezésénél és amennyiben van rá mód, nem ráhelyezni. Szöveget felismerni *OCR (Optical Character Recognition)* megoldásokkal [15] lehet, de esetünkben a szövegi tartalmuk kevésbé lényeges, mintsem pontos helyzetük. Ezért érdemes kevésbé robusztusabb technikákkal is próbálkozni.

Logó és márkajelzések. Számos cég sok pénzt költ arra, hogy márkája látszódjon a különböző vizuális tartalmakon (például reklámok, sportesemény szponzorok, videoklipekben felbukkanó termékek). Felirattal való elfedésük meggátolná céljukat. A problematikát *Logo Detection*-nek hívják és használható megoldásokkal is rendelkezik [16][17].

Különböző objektumok, arcok. Szintén fontos tartalomnak számítanak a képen található objektumok. A mai *Visual* vagy *Image Recognition* technikák [18] már annyira sok objektumot képesek detektálni, hogy egyes képeken már nem is tudunk semleges

területet kijelölni, ahol könnyedén elférne a felirat. Egyik legfontosabb vizuális tartalomként az arcokat emelném ki, mert ezek eltakarása nagyon zavaró tud lenni a videós és képi környezetben egyaránt, míg mondjuk egy gépjármű, vagy élőlény kitakarása kevesebb zavarral jár. Ezt a problémát igyekeznek megoldani a *Face Detection* technikák [19].

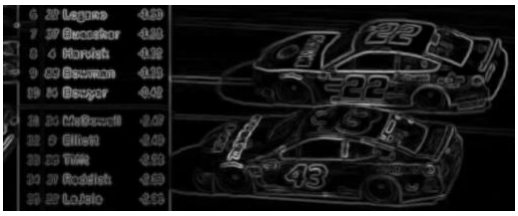
Élek száma és sűrűsége. Egy képnek meghatározó tulajdonsága, a látható élek száma, sűrűsége és pozíciója. Tipikusan élgazdag területek a betűket, szavakat tartalmazó, illetve a fókuszpont közelében megtalálható részek. Általánosságban elmondható, hogy a kép fontosabb részei élesebben jelennek meg, ezért éleik és határvonalaik *éldetekcióval* is könnyebben érzékelhetők [20]. Az éldetekciós módszerek és a témakör az említett szerepe miatt kritikus fontosságú munkámban, ezért már a tervezés során is kiemelt figyelmet kapott a megfelelő algoritmus kiválasztása. Az ezzel kapcsolatos méréseimet a következő fejezetben szedtem össze (3.1.2 fejezet).

3.1.2 Éldetekciós módszerek vizsgálata

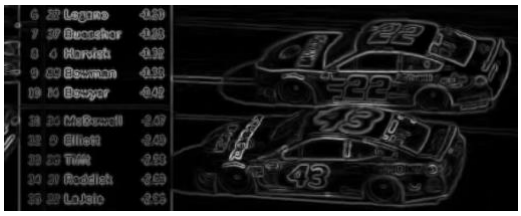
A képi tulajdonságok algoritmikai feltárása időt vesz igénybe. Mivel az építendő rendszer tervezetten videó képein is fog dolgozni, ezért fontos, hogy minél gyorsabban meghatározhatok legyenek a feliratozáshoz szükséges képi paraméterek. Különböző éldetekciós technikák és módszerek ismertek [21], melyek közül szerettem volna kiválasztani a problémám szempontjából legmegfelelőbbet. A *Sobel* két fajta változatát, a *Prewitt*, a *Laplace*, és a *Canny* éldetektorokat vizsgáltam meg részletesebben.

A *Sobel* algoritmus az egyik legelterjedtebb éldetekciós technika, amely egyszerű horizontális és vertikális mátrixműveletek elvégzéséből áll, ezért számítási költsége alacsonynak mondható. A *Sobel* algoritmust többféleképpen is használhatjuk. Ha pontosabb eredményt szeretnénk elérni, akkor érdemes képenként a különböző színcsatornákon (RGB) is elvégezni a mátrixműveleteket. Amennyiben a minőségből tudunk engedni a gyorsaság növelése érdekében, akkor a képet érdemes először szürkeárnyalatossá alakítani és csak egy csatornán elvégezni a számítást.

A bemutatott mintaképen (1.2 ábra) lefuttattam mindkét csatornás megoldást. Az eredményből kivágtam egy lényegesebb részletet és az alábbi ábrákon jelenítettem meg (3.2 ábra és 3.3 ábra ábra). A képeket összehasonlítva láthatjuk, hogy az alsó kocsi versenyszáma (43) a háromcsatornás éldetekció esetén jobban látszódik. A kép egészét vizsgálva azonban nem volt más jelentősebb különbség.



3.2 ábra: Sobel éldetektor eredmény



3.3 ábra: Sobel RGB éldetektor eredmény

A *Prewitt* éldetektáló algoritmus nagyon hasonló az egycsatornás *Sobel*-hez, azzal a különbséggel, hogy más szűrőmátrixot használ a számítás során. A *Prewitt* éldetekcióval készült eredmények világosabbak, mert kevésbé jelennek meg a detektált élek melletti szürke színárnyalatok. Különbség csak nagyfelbontás esetén érzékelhető, detektált élek számában és pozíciójában nincs különbség (ezért ez az eredmény a mi szempontunkból tekinthető azonosnak a *Sobel* eredményeivel).

A *Laplace* operátor jelentősen eltér a már ismertetett algoritmusoktól számítási módban és eredményben is. Az eredeti kép derivatívjaival számol és nagyon érzékeny a képi zajokra is, ezért általában Gauss transzformációt szoktak alkalmazni a *Laplace* előtt. Az eredmény egy alapértelmezetten szürke kép, amin sötét és világosabb színek jelölik az éleket. Erre láthatunk példát az alábbi képen a már fentebb is bemutatott képrészleten futtatva (3.4 ábra).

3.4 ábra: Laplace éldetektor példaeredmény

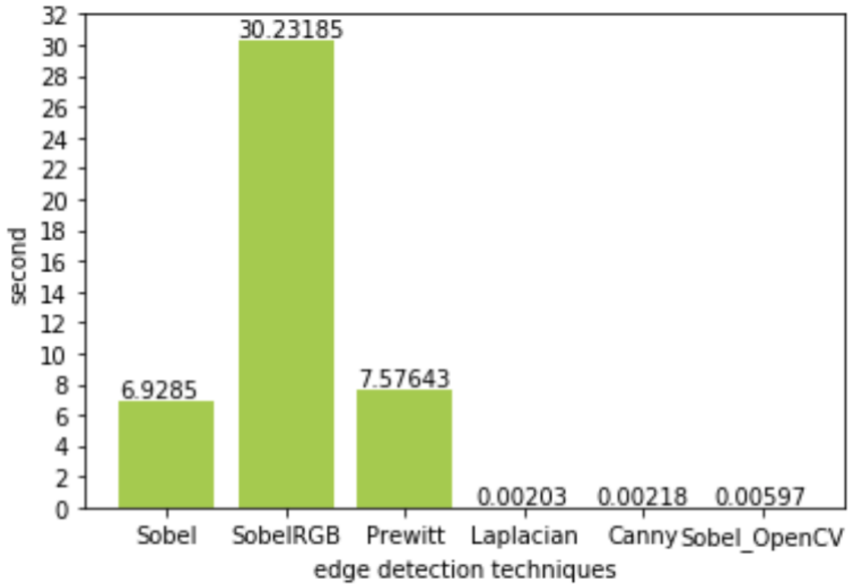
3.5 ábra: Canny éldetektor példaeredmény

A Canny éldetekor a legelterjedtebb és legeffektívebb algoritmusként van számontartva. Számítási mechanizmusa jóval komplexebb, mint az ismertetetteké, amely során Gauss transzformáció után Sobel vagy Prewitt (implementáció függő) detekciót is végre kell hajtani majd végül egyéb zajcsökkentő és élkiemelő módszerekkel kapjuk meg a végeredményt. A megszokott képrészlet Canny éldetektált eredménye látható a fentebbi ábrán (3.5 ábra).

A megfelelő éldetekciós módszer kiválasztása kulcskérdés a feliratpozícionálást tekintve, mivel az élek írják le a legtöbb és legpontosabb információt az ismertetett tulajdonságok közül. A *Laplace* módszer esetében látható, hogy a kimenet egy szürkés

kép, aminek a felhasználása további szükséges egyszerűsítéseket igényelne, ezért ezt a lehetőséget hamar kizártam a további gondolkodásból. A Canny módszernek megvan az az előnye, hogy tisztán jelöli az éleket szinte zajtalanul a Sobel és Prewitt technikákkal szemben. Azonban ez olyan szempontból kevésbé kedvező, hogyha például a pixel színekhez egyfajta prioritást szeretnénk társítani a későbbiek során, ami leírja, hogy az adott pixel milyen fontos része a vizuális tartalomnak, akkor Canny esetén ezt csak binárisan vagyunk képesek megtenni. (Mert csak a fehér-fekete színekkel dolgozik.)

A tárgyalt algoritmusok [21] futási idejét vizsgáltam meg a döntés megkönnyítéséhez. Ehhez a már ábrákon bemutatott képet (1.2 ábra) vettem segítségül 800x443 elbontásban. A forrásban bemutatott implementációk közül csak a Laplace és Canny megoldásoknál használtak OpenCV függvényeket, a többit algoritmus saját kódbázis megvalósításával történt. Az eredményt az alábbi grafikonon vizualizáltam (3.6 ábra).



3.6 ábra: Éldetektorok futási ideje

A grafikont látva egyértelművé vált számomra, hogy nem érdemes kézi implementációra hagyatkozni a Sobel és Prewitt éldetektorok esetében, ezért OpenCV-s megoldással is elvégeztem a detektálást az említett képen. Ennek az eredményét is megjelenítettem a fenti grafikonon (3.6 ábra, jobb szél). Érdekes, hogy a Sobel futási ideje még így is meghaladta a Canny-ét. Ennek az oka valószínűleg a széleskörű igény és törekvés a Canny algoritmus optimalizálására. A számokat és említett tulajdonságokat figyelembe véve a *Canny éldetektort részesítettem előnyben* a munkám további részében.

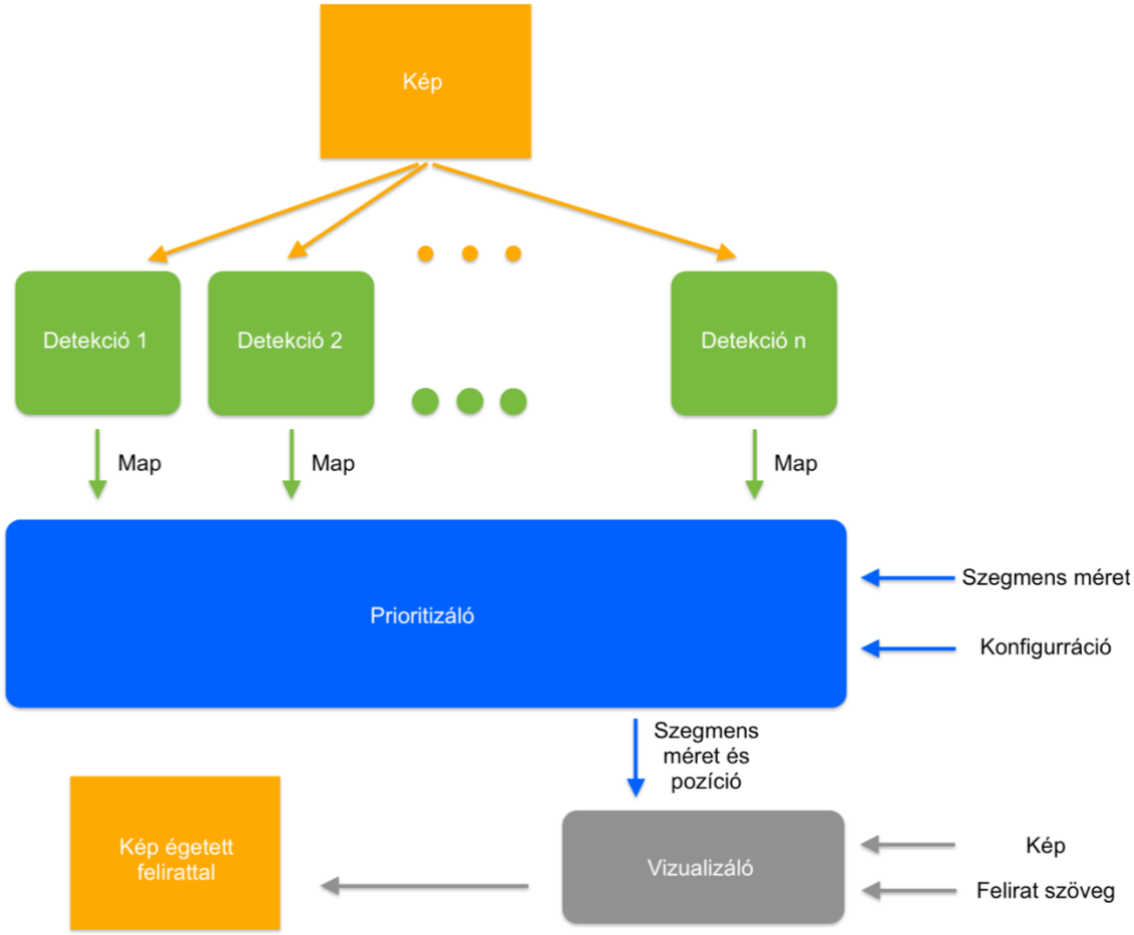
3.1.3 Rendszerarchitektúra

A négy tulajdonság alapvetően határozza meg a tervezendő rendszer szövegillesztési algoritmusát. Szerettem volna olyan megoldást létrehozni, amely nyitott más tulajdonságok későbbi figyelembevételére és ezek integrálása könnyedén elvégezhető. A detekciós algoritmusokat futtató környezetet különálló *komponenseknek* tekintettem, melyek feladata a képekből történő információ kinyerés és ezek alapján egy *megfeleltetési*, vagy másnéven *engedélyező ábra*, *map* létrehozása. Például a szövegdetektáló komponensnek feladata felismerni a szöveget a képen és a hozzá tartozó eredményt könnyen feldolgozható formára hozni egy olyan fájl keretében, ami rámutat a megtalált részletekre.

A komponensek eredményei, azaz a *map*-ek egy *prioritizáló* és *feldolgozó* komponens által kerülnek felhasználásra, amely a megkapott konfiguráció és a szegmens méretének megfelelően keresi meg a legjobb pozíciót. A konfiguráció írja le, hogy milyen tulajdonságokat érdemes figyelembe venni. Vagyis azt, hogy a prioritizálónak milyen detektorokat kell futtatnia a kívánt eredmény eléréséhez. A szegmensméret pedig azt definiálja, hogy mekkora helyet keresünk a képen a felirat részére. A prioritizáló kimenetként egy pozíciót határoz meg, amelyet egy vizualizációs eszköz, vagy komponens könnyedén fel tud dolgozni és ezek alapján a képre égetni. Fontos tehát, hogy a prioritizáló gyakorlatilag nem foglalkozik a felirat szövegi tartalmával, neki a feladata csupán a leghomogénebb terület megtalálása az adott detektorok segítségével. A keresendő terület paraméterei a komponensen kívül kerülnek meghatározásra. Ennek megvan az az előnye, hogy a megoldás más problémára is jól használható és könnyen specializálható a különböző egyedi esetekre, mint például a megfelelő feliratpozíció megtalálása.

Az architektúra tervhez készítettem egy sematikus ábrát (3.7. ábra), amelyen kitűnően végig követhető az adat áramlásának iránya a rendszerben. Látható, hogy a detekciós komponensek megkapják a képet és egy *map* objektumot adnak oda a prioritizálónak. A prioritizáló az említett bemenetek alapján kimentként adja a felirat koordinátákat, amelyből a *vizualizáló* készíti el a végleges képet. Az ábra elsősorban az adatfolyam irányát jeleníti meg. Az egyszerű vezérelhetőség érdekében a prioritizáló vagy másnéven *feldolgozó komponens* felelősségi körébe terveztem a detektorok

futtatását is. Így nem kell minden detektort külön paranccsal elindítani és inputként ugyanazt a képet megadni, de erre az ábrán nem hívtam fel a figyelmet.



3.7. ábra: Képfeliratszegmens pozícionáló rendszer architektúra

3.2 Videó feliratpozícionáló rendszer tervezése

A fentebb bemutatott rendszer segítségével képesek vagyunk megtalálni egy képen a legkevésbé tartalomgazdag területet, ahova a feliratot elhelyezve minimalizáljuk a letakart információt. A bemutatott rendszerre egységként tekintve elmondható, hogy inputként vár egy képet, egy szegmens méretet, konfigurációt, illetve a felirat szövegét. Outputként pedig egy módosított képet kapunk, amire égetve a kívánt pozícióban jelenik meg a felirat. Ahhoz, hogy az algoritmust videós környezetben tudjuk használni az inputok előállítását és az outputok kezelését kell jól meghatároznunk. Amennyiben azt szeretnénk, hogy ugyanaz a feliratszegmens ne tűnjön el és jelenlen meg újra és újra a videó lejátszása alatt, a videó *minden* képére rá kellene égetnünk a feliratot. Ez azon kívül, hogy egy jelentősen költséges feladat, sokszor felesleges is a mi szempontunkból, ha a

kép változása minimális és a felirat textuális tartalma sem változik. Továbbá a feliratégetésre léteznek jól működő, gyors megoldások is. Tehát amíg képek felirat pozícionálására kifejezetten szerencsés az *Open Captioning* megközelítés a képre égetéses funkciójával, addig videós esetben a *Closed Captioing* nagyobb előnyökkel járna. Ugyanis, ha a felirattal égetett videó helyett egy feliratfájl a kimenet (például.: *ass* formátumban), amely jól leírja a feliratok pozícióját, akkor az égetést már könnyen elvégezhetjük egy megfelelő videó kezelő programmal (pl.: VLC [8]).

Az ismertetett képfeldolgozó rendszert alapul véve könnyen módosíthatunk feliratégetésen, csupán a *Vizualizáló* komponenst kell a végéről elvenni és a képfeldolgozó máris a legjobb pozíciókat ajánlja a feliratozott képek helyett. A következő fejezetekben a képfeliratozó, vagy másnéven prioritizáló komponens inputjainak a tervezett előállítási módját, illetve az outputjának kezelését taglalom videós környezetben.

3.2.1 Inputok előállítása a videófeldolgozás során

A videó feldolgozó rendszer fontos alrendszerként fogja használni a képi feliratpozícionáló komponenst, ezért elő kell állítani a képfeldolgozáshoz szükséges inputokat. Ez egy konfigurációt, egy feliratszegmensméretet és magukat a képeket jelenti.

A ***konfiguráció*** beállítás leírja, hogy milyen detektorokat alkalmazzunk az adott kép elemzéséhez, és azokat milyen prioritással kezeljük. Ez a paraméter videó feliratpozícionálásnál is könnyen értelmezhető és hasznos, ha kívülről is vezérelhető a videó tartalmától függően. Ezért érdemes a videófeldolgozó rendszer inputjára is kivezetni és a felhasználóra bízni, hogy milyen tulajdonságokra szeretne optimalizálni a pozícionálás során.

A ***feliratszegmens méret***ét azonban már kevésbé célszerű a felhasználóra bízni. A különböző felbontású videókra külön méretet kell számolni és ez nem tenné elég felhasználóbaráttá a tervezendő rendszert. Szerettem volna a megoldást úgy megépíteni, hogy az könnyen futtatható legyen egy hozzá kevésbé értő szereplő számára is és meg akartam spórolni a videó és felirat méreteinek számolását. Ezért már tervezésnél ügyeltem rá, hogy a méretek számítása a videófeldolgozó modul komplexitásához tartozzon és ne kívülről jövő információ legyen. A méret számításához ismerni kell a pontos szöveget, hogy kalkulálhatóvá válhasson a szövegblokk pixelekben mért hossza és sorainak száma a használt betűtípus és betűméret alapján.

Tehát a szövegi tartalomra szükség van a számítások elvégzéséhez az építendő rendszernek. Mivel a feliratokfájlok legelterjedtebb formátuma az *SRT,* ezért úgy döntöttem, hogy a videófeliratpozícionáló megoldás inputjai közé felveszem a szövegi információkat leíró *SRT* fájlt is. A fájlból könnyen felolvashatók futás során a szegmenssorok és kiszámításra kerülhetnek a feliratméret értékek. Mindemelett *SRT* fájlok használatával a birtokunkba kerülnek a szövegblokkok időzítési adatai is, amire szintén szükség lesz a videófeldolgozás során.

SRT fájlok előállítása azonban kérdéses lehet egyes videóknál, ahol ezek nem állnak alapvetően rendelkezésre. Ennek a megoldására ahogy már említettem (az 1.1.2-es fejezetben) számos szolgáltatás létezik, melyeket fel lehet használni *SRT* generálásra. Én az IBM Watson Media fejlesztését, a *Video Enrichment*-et [22] használtam tesztvideók feliratozására, mert már nagy tapasztalattal rendelkeztem a terméket illetően.

Input ***képek*** előállítása a videó képkockákra vágásával könnyen megoldható. A leírtak alapján felesleges minden képkockán végigmenni és rájuk szöveget égetni. Ezért meg van a lehetőségünk, hogy csak azokon a képkockákon futtassunk pozíciószámítást, amelyeket valamilyen szempontból értékesnek tartunk. A legfontosabb kérdés tehát input képek előállításához, hogy melyik képeket ítéljük szükségesnek a feliratok pozícionálásához? A feliratszegmensek időről időre változnak, de akár több másodpercig is láthatóak maradnak. Minden szegmenshez tartozik valamennyi képkocka, amelyeken a felirat megjelenik. Számítási időt tekintve minimális megoldást jelentene minden feliratszegmens változásnál egyszer lefuttatni a feliratpozícionálót és az aktuális legjobb helyzetbe illeszteni a feliratot. A maximális megoldás pedig az algoritmus képkockánkénti futtatása lenne szegmensenként külön-külön. Értelemszerűen a maximálás megoldás jóval hosszabb időbe és kapacitásba telik amellett, hogy használatával a felirat helyzete képkockáról képkockára teljesen megváltozhatna, amely a felirat „ugrálását” jelentené. Ezt a nagyon zavaró jelenséget szerettem volna elkerülni.

A maximális és minimális megoldás között a lehetőségek száma számos. Nem mindegy, hogy a szegmensek milyen ütemben változtatják a pozíciójukat, melynek helyes beállítása sokkal inkább szubjektív tényező, mintsem objektív méréseken alapuló döntés kérdése. Mindenképpen szerettem volna több opciót is kipróbálni és megvizsgálni, hogy milyen élményt nyújtanak tartalomfogyasztás szempontjából. Három csoportot és ezzel együtt algoritmust határoztam meg. Egy sok mintavételen és pozíciószámításon alapuló ***dinamikus*** megoldást, egy feliratszegmenseket tekintve ***stabil*** pozíciókkal rendelkező

számítást és egy, a videó hossza alatt ***fix*** elhelyezéssel rendelkezőt. Mindhárom technikának van előnye és hátránya, melyeket a könnyebb átláthatóság kedvéért az alábbi táblázatban foglaltam össze (2. táblázat).

Dinamikus pozícionálás	
Előnyök	- A feliratszegmensek dinamikusan alkalmazkodnak a képi megjelenéshez. - A legkevésbé zavaró hely könnyen megtalálható az egyes képkockákon külön-külön, így a feliratozott videóban összességében is kevesebb vagy teljesen elkerülhető a lényeges tartalom kitakarása.
Hátrányok	- A folyamatos mozgás nehezíti az olvashatóságot. - A kimeneti formátum bonyolultabb. Egy feliratblokkhoz, több pozíció is tartozik. - Számítása több mintavételt és időt igényel.
Stabil pozícionálás	
Előnyök	- Pozícióváltozás csak szegmensenként történik, az egyes blokkok helyzete stabil. - A feliratblokkok a legoptimálisabb pozícióba kerülnek. - Kevesebb mintavételt és gyorsabb számítást igényel - Egyszerűbb a kimeneti feliratfájl formátuma. Egy blokkhoz egy pozíció tartozik.
Hátrányok	- A blokkonként változó helyzet zavaró lehet - A szegmens alatti vizuális tartalomváltozást nem követi le a felirat pozíciója
Fix pozícionálás	
Előnyök	- Felirat fix pozícióban, nincs semmilyen mozgás. - Kevesebb mintavétel, gyorsabb számítás. - Egyszerű kimeneti fájl, egy fix pozíció az összes blokkot tekintve
Hátrányok	- A felirat nem vált pozíciót, ezért kitakarhat lényeges információt

2. táblázat: Pozícionáló algoritmusok összehasonlítása

Fontos kiemelni, hogy a stabil pozícionálás esetén is változnak a feliratszegmens pozíciók, de egy szegmenshez egy pozíció tartozik, míg dinamikus esetben feliratszegmensen belüli helyzetváltozásokkal kell számolnunk.

A táblázat (2. táblázat) alapján elmondható, hogy azokban az esetekben érdemes fix pozícionálást választani, mikor nagyobb hangsúly van a felirat megértésén és a szövegi tartalom fontosabb, mint a képi/vizuális. Ezzel szemben a dinamikus pozícionálás nagyobb hangsúlyt fektet a tökéletes, legkevésbé zavaró helyzet kiválasztására, így nagyobb vizuális élményt adó videóknál jelenthet előnyöket.

Tehát most már tisztában vagyunk a fix, a stabil és dinamikus felirat pozícionálás előnyeivel és hátrányaival. Érdemes megnézni, hogy tervezés szempontjából algoritmikailag miben különböznek egymástól. A stabil pozícionálás feliratszegmensenként találja meg a legjobb pozíciót, ezért a képkockákon történő detekció futtatások számát a szegmensek száma befolyásolja. Így nevezhető a stabil pozícionálás szegmens alapúnak. Ezzel szemben a dinamikus és fix pozícionálás idő alapú, a detekciók adott időtartamonként futnak és az algoritmus számításának idejét a videóban szereplő feliratozott percek befolyásolják.

Diplomatervemben szerettem volna mindhárom algoritmussal mélyrehatóan foglalkozni és később kiértékelni őket használhatóságuk alapján. A különböző algoritmusok alkalmazásával előállíthatóak a képfeldolgozó komponens számára az input képek is, ezzel lefedve az összes szükséges inputot.

3.2.2 Output kezelése

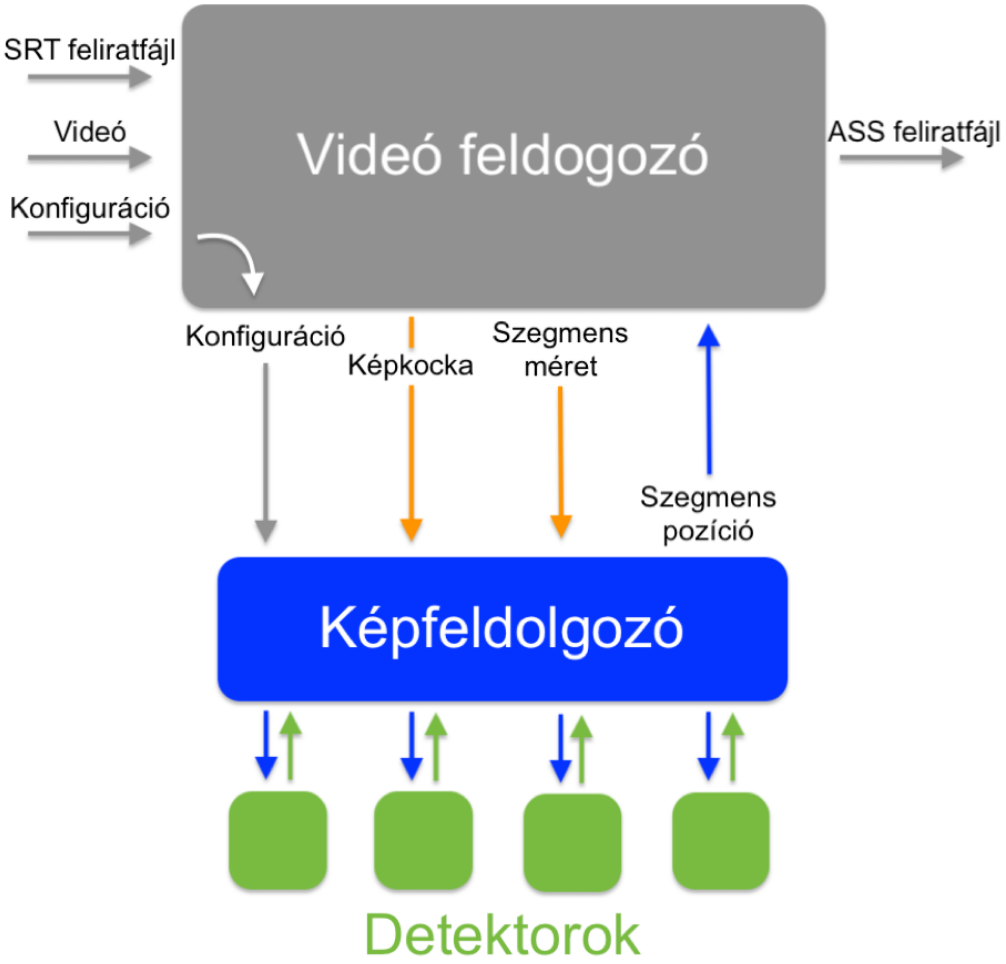
A képfeldolgozó komponens outputja egy koordináta a felirat legjobb pozícióját illetően. Ennek a pozíciónak az értelmezése és kezelése nagyban függ a választott pozícionáló algoritmustól. Bizonyos helyzetekben érdemes lehet őket összegyűjteni és stabilizálni, hogy csökkentsük a zavaró ugrálást. Ezért koordináták kezelésével kapcsolatos pontos részleteket a bemutatott algoritmusok megvalósításánál ismertetem (4.2 fejezet).

Alaposan elmerülve a videó feliratpozícionáló algoritmus működésének tervezésében a legcélszerűbb megoldásként az úgynevezett Advanced SubStation Alpha (*ASS)* fájlok [23] előállítását láttam. Ezzel a formátummal le lehet írni szegmensenként a megjelenő felirat helyzetét. Könnyen olvasható emberi szemmel is és a legelterjedtebb fájlformátumként tartják számon a részletesebb feliratfájlok között, de népszerűsége még

így is jóval alacsonyabb az *SRT*-hez képest. Számos videólejátszó nem támogatja a kezelését, de szerencsére a VLC [8] képes az *ASS* formátum használatára, így szükség esetén a keletkezet fájl VLC-vel ráégethető a videóra és könnyen megvalósítható a tervezett égetett videófájl.

3.2.3 Videó feliratpozícionáló architektúra

Az alábbi ábrán összefoglaltam a bemutatott architekturális képet (3.8 ábra). Jól látszódik a rendszer három bemenete bal oldalt *(SRT, Videó, Konfiguráció)* és a kimeneti *ASS feliratfájl*. A konfiguráció egy része átadásra kerül a képi feldolgozó egységnek, a videókból képkockák készülnek a feliratfájlból pedig szegmens méretek kerülnek meghatározásra a Videó feldolgozó komponens keretében. Látható, hogy a korábbi ábra (3.7. ábra) Prioritizáló egysége újra hasznosításra kerül *Képfeldolgozó* egységként, amely a prioritizálás mellett ellátja a detektáló algoritmusok kezelését is. A megkapott konfigurációból kiolvassa, hogy milyen detektáló algoritmusokat kell elindítania az adott képkockán és ezeket lefuttatja az inputján a szintén megkapott szegmens méretet figyelembe véve. Válaszként a szegmens pozíciót határozza meg a *Videó feldolgozó* számára. Amely feldolgozza a választ és elvégzi a szükséges utószámításokat (például koordináta rendszer transzformációk az adatok között, *ASS* fájl előállítás, szükség esetén feliratpozíció interpoláció stb.). A feldolgozás végén előáll az *ASS* fájl, ami könnyen a videóba égethető más programok segítségével.



3.8 ábra: Videó feliratpozícionáló rendszer architektúra

4 Megvalósítás

Ahogy már az előző 3 Tervezés fejezetben is két alrészre osztottam a rendszer bemutatását, jelen fejezetben is folytatom ezt a bontást. Az első rész (4.1) a képeken történő feliratpozícionáló rendszer megvalósításáról taglalja, míg a második (4.2) részletezi a felhasználását és egyéb logika implementálását a videófeliratpozícionálást elvégző rendszerhez.

A programot Python nyelven készítettem el. A videó feldolgozáshoz és a különböző detektorok implementálásához az OpenCV programkönyvtárat használtam [26]. A képekkel történő munkához a Pillow könyvtár [27] algoritmusait választottam.

4.1 Képfeliratpozícionáló rendszer megvalósítás

A detektáló komponenseket egységbe zártan közelítettem meg. Mindegyik detektor bemenete egy kép és kimenete egy *map,* ami leírja a fontosnak tartott részeket, egy fekete-fehér szürkeárnyalatos kép formájában. A komponensek bemutatása prioritási sorrendben olvasható. Legfontosabb az éldetektor, mert univerzálisabban használható a többinél a képek típusától függetlenül.

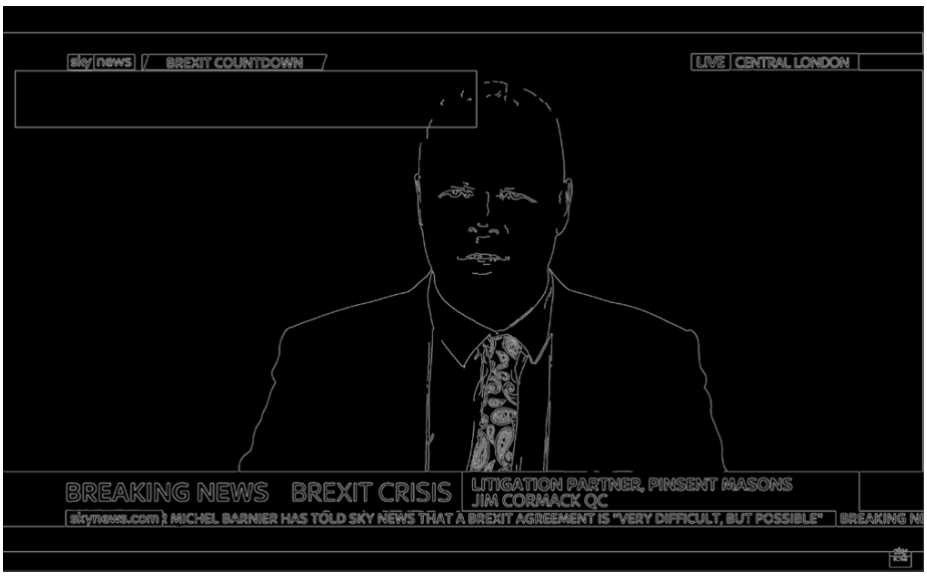
4.1.1 Éldetekció

Egy kép lényegi tartalmát jól összefoglalják a látható objektumok élei. Az éleket önmagukban vizsgálva is már egy jó közelítést kaphatunk az ajánlott feliratpozícióhoz. Emberi szemmel akár csak az élek rajzolatából is felismerjük, hogy egy márkát, szöveget, vagy más objektumot ábrázol az eredeti kép. Ebből a szempontból ez a technika egy általánosításnak tekinthető a másik háromhoz képest. Viszont bizonyos esetekben kizárólagos használata nem kívánt eredményhez vezetne. Például egy olyan képen, ahol a lényeges tartalom egyszerűbb mintázatú és kevesebb élt tartalmaz, mint a háttér vagy más képterület.

Többfajta algoritmus is létezik éldetekciós problémamegoldásra, melyeket már a kapcsolódó fejezetben (3.1.2) részletesen bemutattam. A *John F. Canny* által kidolgozott eljárás mellett döntöttem, amely képes az élek erőteljes elhatárolására, így két színnel (fehér-fekete) leírható az adott éldetektált kép. A Canny edge detektor számításai közben több állapottal rendelkezi és képes érzékelni élek erősségének széles skáláját. Az

állapotok különböző képtranszformációs műveletek határait jelölik (például zaj csökkentés, gradiens meghatározás stb.). Továbbá kiemelendő, hogy az algoritmus szürkeárnyalatos képeken működik, ezért előzetes átskálázást kell végezni színes képeken [24].

Mivel szerettem volna mielőbb látni ötletem eredményességét, ezért az algoritmus alkalmazásával elkészítettem egy éldetekción alapuló feliratelhelyező programot, amely képes volt a feliratot azonosító szövegdobozt pozícionálni az éldetektált képen. A komponensbe zárást és az elvárt ki- és bemeneteknek megfelelő működést csak később végeztem el. Működés során a képekből először szürkeárnyalatos változat készült, majd az élek detektálása következett. Az így kapott eredményen egy élminimumot kereső implementációt készítettem, amely képes detektálni azt a területet, amely a felirat szövegdobozának méretével rendelkezik és a legkevesebb élt tartalmazza.



4.1. ábra: Éldetekció alapján készített feliratpozíció ajánlás

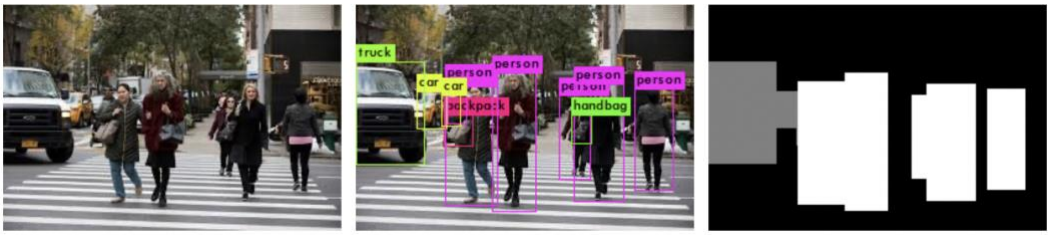
A kapcsolódó ábrán (4.1. ábra) jól látható az elkészített program vizualizált eredménye. A már bemutatott ábrához (3.1 ábra) hasonló kiindulási képből egy letisztított, éleket tartalmazó kép készült, majd egy fehér keretes szövegdoboz került elhelyezésre a legkevesebb élt tartalmazó részen. Az ábrán (4.1. ábra) pontosan látszódik a kizárólagos éldetekció használat hátránya. Ugyanis a képen szereplő arc fejtetője tartalmazta a legkevésbé sűrűbb területet, így a szövegdoboz rálóg a személy fejére, ami egy videó nézése közben igen zavaró lenne. Ez a probléma könnyen javítható, ha a megtalált élek mellett más tulajdonságokat is figyelembe tudnánk venni egy arc, vagy objektumdetektálóval, amelyet a következő fejezetben mutatok be.

4.1.2 Objektumdetektálás

Feliratszegmensek okos elhelyezéséhez meg kell értenünk, hogy mi, vagy milyen objektumok szerepelnek a képen. Objektumdetekciót képeken már nagyfokú hatékonysággal lehet folytatni mesterséges intelligencia alapú számítógépes technikákkal. Számos szoftver és algoritmus áll rendelkezésre, melyek képesek ezt a problémát megoldani. A legismertebb technikák neurális hálózat alapúak, ezért betanítást és sok optimalizálást igényelnek. Mivel mesterséges intelligencia alapú megoldást szerettem volna nyújtani a feladatomra, ezért mindenképp egy hálózat alapú algoritmusban gondolkoztam. Választásom a YOLO (*You only Look Once)* nevezetű architektúrára esett, mert számos felmérésen kiemelkedő helyen szerepel és elsősorban ezt ajánlják objektumfelismerésre [28]. További nagy pozitívuma, hogy valós idejű működésre is képes, ami videók feldolgozásánál hasznos tud lenni [29].

YOLO használatánál lehetőségünk van a hálózat újra tanítására különlegesebb adathalmazok esetén, de alkalmazhatunk már előre feltanított modelleket is. Az én problémámat szerettem volna minél széleskörűbben megoldani és nem egy adott videótípusra optimalizálni, ezért úgy döntöttem, hogy nem merülök el egy megfelelő adathalmaz keresésében és felcímkézésében. Az alapértelmezett paraméterezés jól használható általánosabb objektumok felismeréséhez (pl.: emberek, autók, tárgyak), melyet alkalmasnak véltem a problémámhoz. Más beállításokkal bizonyos témakörben hatékonyabb, de általánosságban pontatlanabb megoldás készült volna. Ebbe az irányba akkor érdemes tovább vizsgálódni, ha már tudjuk előre, hogy a feliratozandó videó milyen vizuális témakört érint, így a tartalom szempontjából értékes objektumok hatékonyabban lokalizálhatók. Erre az opcióra továbbfejlesztési lehetőségként tekintettem, nem pedig megvalósítandó részfeladatként az említett általános felismerési szándék következtében.

Az elkészült programom a fentiek alapján mesterséges intelligencia használatával a képeket áttranszformálta, egy olyan megjelenítésbe, amely leírja, hogy a pixelek milyen fontosságúak a feliratra nézve. Tehát a bemeneti képen először lefuttattam az objektum felismerést, majd a megtalált objektumokat osztályoztam a fontosságuk alapján. Az emberi alakoknak adtam alapértelmezetten a legnagyobb prioritást, ezért ők kapták a legvilágosabb színezetet. A többi objektumot egy szinttel kevésbé fontosnak klasszifikáltam, ezért a kimeneti *map*-en ezek már szürke színnel szerepelnek. A folyamat eredményét szemlélteti a 4.2. ábra.



4.2. ábra: Objektumdetekciós eredmények felhasználása

A program bemenete egy egyszerű kép (4.2. ábra bal oldala), amin a YOLO segítségével objektumdetekció történik (középső kép). Az eredmény egy szürkeárnyalatú megfeleltetés (jobb oldal), amely feketének jelöli az objektumok szempontjából lényegtelennek tartott részeket és fehérrel a legfontosabbakat. A kettő közötti árnyalatok pedig a köztes észrevételeket. A technikának az az előnye, hogy így emberi szemmel és számítógéppel is könnyen emészthető a felismerés eredménye és jól leírja az objektumok egymás közötti relációját. A fontosabb tárgyak kitakarhatják a kevésbé fontosabbakat, ezért lehetséges, hogy a középső képen zölden látszódó *handbag* a végeredményben már nem tűnik fel. Az eredményt felhasználva a komponenseket integráló program sokkal könnyebben megtudja határozni, hogy melyik objektumot a legkisebb probléma kitakarni, amennyiben nincs másra lehetőség.

4.1.3 Karakterlokalizáció

Ahogy az eddigi ábrákon is látszódott a karakterek és megjelenített szövegek fontos szerepet játszanak videókon és képeken. Feliratot helyezni egy már meglévő szövegre vagy feliratra különösen zavaró mert, nem csak a meglévő karaktereket teszi olvashatatlanná, a ráhelyezett szegmens elolvasása is nehézkessé válik. Karakterek felismerésével az *OCR* problematika foglalkozik, mely válaszolhatna arra a kérdésre, hogy hol helyezkednek el a szövegek a képen, de számítási ideje és komplexitása bőven meghaladja a számomra szükségeset. Az én esetemben nem fontos, hogy a fellelhető szövegek mit írnak le, vagy milyen nyelven láthatóak, egyedül csak a helyzetük lényeges. Karakterek lokalizációjára, felismerés nélkül lényegesen gyorsabb és egyszerűbb megoldások állnak rendelkezésre, amelyek elegendőek célomra.

2017-ben Xinyu Zhou és csapata publikálta a röviden *EAST* nevezetű neurális háló architektúrát, amely gyorsan képes a karakterek lokalizációjára és könnyen felhasználható más problémákra is [30]. Én az általuk készített modellt használtam fel a szövegdetektálásra alkalmas mesterséges intelligencia alapú programban [31].

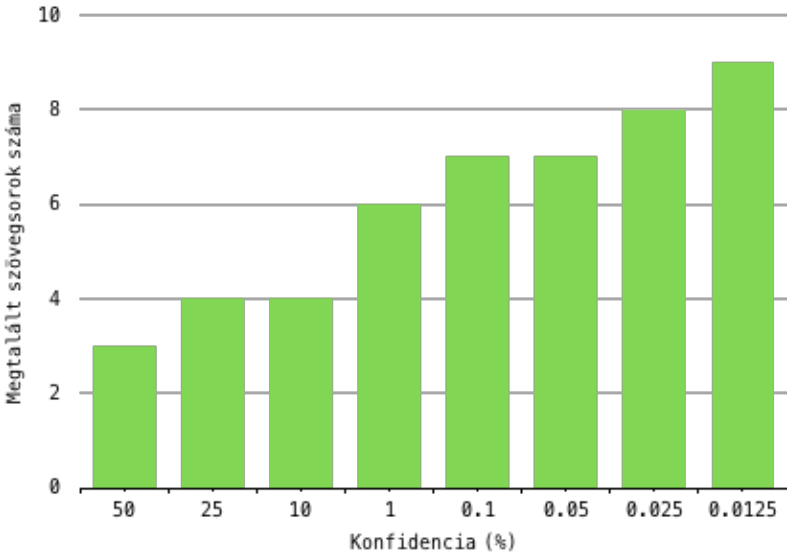
A megírt program feladata hasonlóan az objektumdetektálóéhoz (4.1.2 fejezet) csupán annyi, hogy kijelölje a fontosabb részeket a képen. Ebben az esetben nem láttam értelmét prioritásos sorrend felállításának, mert a szöveg mérete, helyzete nem feltétlen írja le fontosságát.

A szövegdetektáló algoritmus úgy működik, hogy egy konfidencia számot rendel a felismert képrészletekhez. Azt, hogy mit detektál karkatereknek a konfidencia állításával tudjuk befolyásolni. Az érték finomhangolásához többek között a már bemutatott két képet is felhasználtam (1.2 ábra és 3.1 ábra). Először 50%-os konfidenciaszint felett tekintettem egy képrészletet szövegnek, ami a híradós képnél (3.1 ábra) teljesen megfelelő volt. Megtalálta az algoritmus az alul-felül látható karkatereket. De az autóverseny részletnél látható szövegeket kis pontossággal találta meg (4.3. ábra, felső kép).

4.3. ábra: Karakterdetektálás a mintaképeken 50%-os és 0.01%-os konfidenciaszinttel

50%-os konfidenciával látható, hogy a képen szereplő baloldali listán csak három nevet tudott detektálni. A százalékot szisztematikusan csökkentettem, melyet az alábbi

diagrammon lehet szemmel követni (4.4. ábra). A diagramm függőleges tengelyén a fenti ábra (4.3. ábra) bal oldalán található 20 elemű névsor felismert szövegsorainak számát látjuk. Vízszintes tengely a vizsgált konfidencia százalékokat azonosítja. 50%-nál a megtalált sorok száma három, míg 0.0125%-nál kilenc, de itt már tévesen is szövegként detektált egy nem releváns képrészletet (4.3. ábra, alsó kép). Sajnos látható, hogy ilyen alacsony érték mellett sem vett észre minden szöveget. A baloldali névsor felét így sem ismerte fel. A további csökkentés már erős negatív hatással lenne az eredményre, ezért a konfidenciát 0,025%-on hagytam, mivel ezzel az értékkel találta meg a legtöbb szöveget úgy, hogy más vizuális elemet, nem kategorizált karakternek.



4.4. ábra: Megtalált szövegsorok száma a konfidencia függvényében

A program kimente hasonló az objektumdetektálásnál (4.1.2 fejezet) már ismertetett formátumhoz azzal a különbséggel, hogy csak fekete és fehér színeket használtam információhordozóként. Ezzel a technikával a modulok kimenete könnyen összekapcsolható, ha képpontonként egy logikai VAGY műveletet végzünk. A keletkezett fehér színnek megfelelő egyesek (vagy nem nullások) leírják a pixel fontossági értékét.

4.1.4 Logó és márkajelzés detekció

Logók és márkajelzések detektálása nem csak feliratozási szempontból érdekes. Reklámkampányok eredményessége jól mérhető a megjelenített márkajelzések számával, így a logó detekcióban a különböző kis és nagyvállalatok is érdekeltek. Továbbá

szükséges bizonyos esetekben a logók kitakarása, ha a hozzátartozó vállalat nem egyezett bele márkajelzésük feltüntetésébe.

A témakör nehézsége az, hogy robosztus megoldás, ami képes több száz márkát felismerni, nagyon sok tanítómintát igényel. A különböző felismerendő logók képeiből külön-külön. Mivel ilyen megoldás összeállítása igen nehéz és szabad felhasználású változata nem is beszerezhető, ezért egy nagy hatékonyságú logódetektorról le kellett mondanom. Nagy szakirodalommal rendelkezik a témakör és könnyű elmélyülni benne. Általában hálózatfeltanításról és konfigurálásról számolnak be a források [32]. Én szerettem volna egy olyan megoldást használni, ami egyszerűen beilleszthető eddigi munkámba és nem igényel hosszas környezetkialakítást. Végül a YOLO hálózatarchitektúra mellett döntöttem, melyhez találtam olyan súlyokat és konfigurációt, amely márkajelzések detektálásához készült [33]. A paraméterezés 47 márkajelzést képes felismerni melyek között szerepelnek nagy világmárkák is (pl.: Ford, Nvidia, CocaCola stb.) [34]. A következő képen (4.5. ábra) látszódik, hogy a megjelenített 7 márka közül csak 2-t ismer fel az algoritmus. Az Apple logójára is fel van tanítva a hálózat, azonban ebben a formában nem tudta beazonosítani, csak a Pepsi és a Google márkákat. Más képeken a harapott alma jelet gond nélkül megtalálta. Mivel a tématerület igen nagy és javulást hosszas elmélyüléssel lehetne elérni ezért a gyors korrekcióra nem volt lehetőségem.

4.5. ábra: Logódetekció eredménye, ahol a 7 márka közül a színesen keretezetteket találta meg az algoritmus.

A hálózatot futtató algoritmusom azonos koncepció alapján lett elkészítve, mint a karakterdetekciós eljárás. Így mesterséges intelligencia használatával a bemeneti kép eredménye egy ugyanolyan fekete-fehér megfeleltetése volt a képnek, melyből könnyen ki lehet nyerni a fontos képrészleteket algoritmikailag és emberi szemmel is.

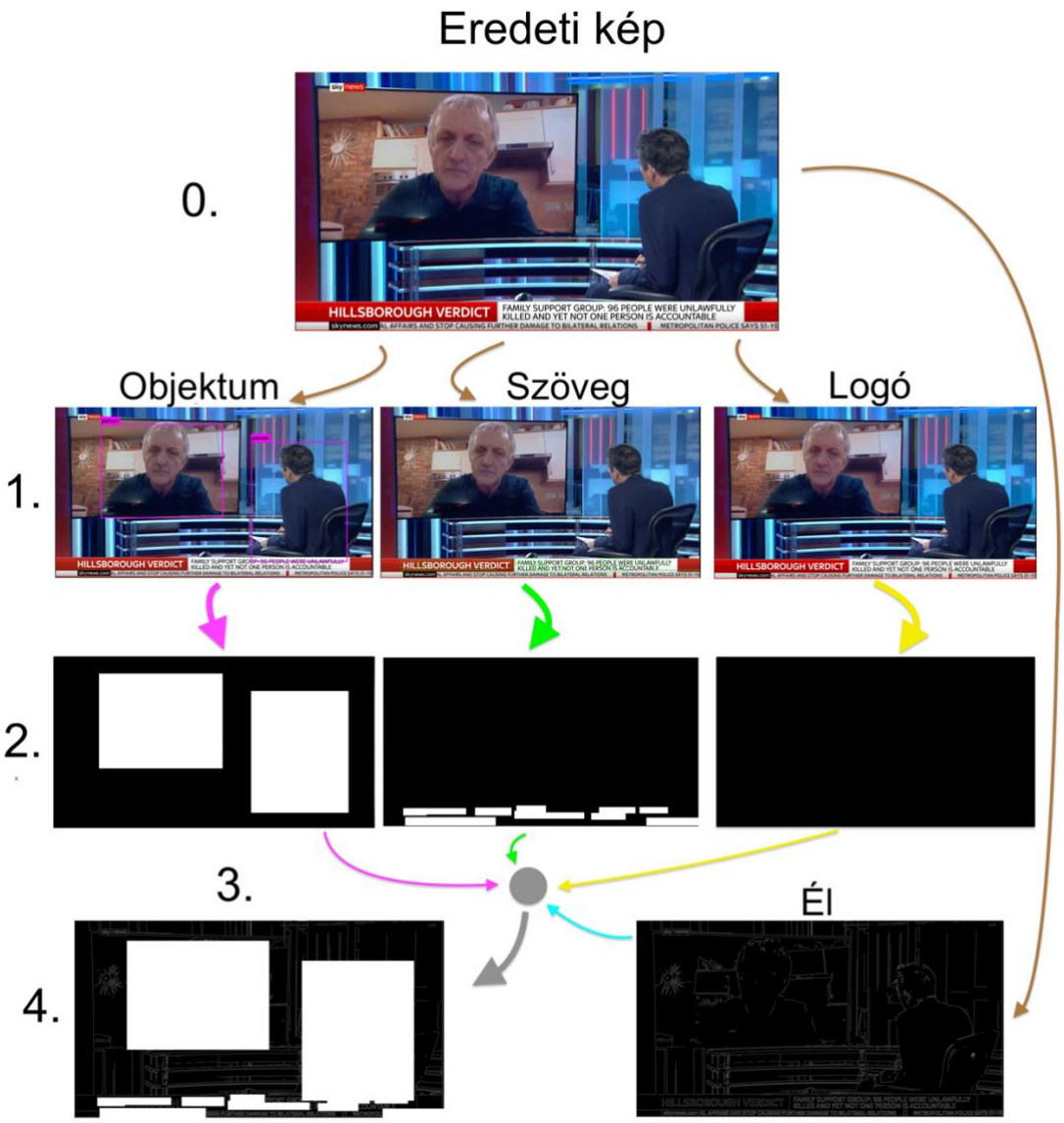
4.1.5 Prioritizáló komponens

Tehát az ismertetett képtulajdonságokat négy különféle módszerrel észleltem és dolgoztam fel. A következő lépés az eredmények felhasználása, amely ahogy az architektúrális ábrán is jelöltem (3.7. ábra) a prioritizáló komponens hatásköre. Szerettem volna a rendszert könnyen használhatóvá tenni, ezért ennek a komponensnek a határkörébe implementáltam a különböző detektáló algoritmusok futtatását is. Ennek nagy előnye, hogy az algoritmusokat nem kell egyesével elindítani és eredményüket egy közös helyen elérhetővé tenni, hanem a prioritizáló komponens ellátja ezt a feladatot is. Továbbá a videófeldolgozáshoz való felkészüléshez is hasznos ez a megközelítés.

A komponens az inputként megadott konfigurációja alapján indítja el a szükséges detektáló algoritmusokat párhuzamosan. Mivel az algoritmusok esetenként hosszabb időt igényelnek ezért fontosnak láttam a párhuzamos működésmód támogatását a prioritizáló komponensen belül is. Miután a felkonfigurált algoritmusok futása lezajlott, a detekciós eredmények összesítésre kerülnek.

A feldolgozási mechanizmust szemlélteti az alábbi ábra (4.6. ábra) egy tetszőlegesen választott SkyNews híradójából kiragadott képpel. 0-ik lépésben a híradó kiindulási képe látható. Első lépésben kiemelésre kerültek a beazonosított elemek (objektum, szöveg és logó). A SkyNews logója is szerepel a képen, de ezt a logódetektor sajnos nem képes beazonosítani, ezért a hozzá tartozó *map* fekete marad. Azonban a szöveg és objektum felismerő sikeresen azonosítja a látható elemeket. A második sorban vizualizálásra kerültek a detektált területek fehér lenyomatai. Harmadik lépésben, a három kizárásos kép egyesítésre kerül az éldetektor eredményei mellett. Ezáltal előáll a *prioritizált* kép.

A kialakult prioritizált képet többféleképpen kezelhetjük a lenti ábrán (4.6. ábra) azt a verziót láthatjuk, mikor a képi tulajdonságok nincsenek befolyással egymásra. Mindegyik tulajdonság ugyanolyan fehér jelölést eredményez az összesített képen. Eggyel precízebb megoldás, mikor preferencia sorrendet állítunk a képi tulajdonságok között és ezalapján hozzuk létre a prioritizált képet. Ez például olyan esetekben lehet hasznos, mikor előre tudjuk, hogy a vizuális tartalom során sok lesz a szöveg és mindenképpen el szeretnénk kerülni azokat a helyzeteket, amikor föléjük kerül a felirat.



4.6. ábra: Képfeldolgozás bemutatása

Egy általánosabb és hasznos megközelítés, mikor az összegzett képen/mátrixon normalizációt hajtunk végre. A normalizáció segítségével a képpontok értékei 0-1 tartományba kerülnek, ami azt jelenti, hogy azok a képpontok, amelyek több detektor szerint is hordoznak fontos információt fehérebbek lesznek a többinél. Ez egy emberi szemmel is jól látható köztes eredményt jelent a komponensben. Normalizációt használó prioritizált kép előállítására a következő ábrán látunk példát (4.7 ábra). Az ábrán látható eredeti kép már ismerős lehet az 1.2 ábra képéről.

Tehát a prioritizáló komponens a megkapott inputokat összesíti egy képre, melyen ezáltal megjelennek a fontos és kevésbé fontos területek az élek mellett és kialakul a *prioritizált* kép. Az így kapott eredményen egy csúszó ablak segítségével végig iteráltam.

A csúszó ablak területe megfelel a megadott felirat szövegdoboz méretének. Erre a területre eső sűrűséget számoltam ki ciklikusan, ami gyakorlatilag a nem fekete képpontokhoz tartozó értékek összegének felel meg. Mivel nagyobb felbontású képeken is szerettem volna használni az algoritmust, ezért a képpontonkénti iterálás és számításvégzés hosszas ideig tartott volna. Az elhúzódó számítási időt, úgy küszöböltem ki, hogy az egyes iterációkhoz tartozó növekményt állíthatóvá tettem.

4.7 ábra: normalizációs módszerrel detektált képi tulajdonságok

4.8. ábra: Az újra pozícionált felirat elhelyezve a képen

A megtervezett rendszer algoritmusa alapján készült el a fenti ábrán látható feliratozás (4.8. ábra). A definiált képtulajdonságok figyelembevételével a legkevésbé zavaróbb helyre került a felirat. Ezt zölden bekeretezve jelöltem. Közelebbről megnézve észrevehető, hogy a felirat szövegdobozának bal sarka beleesik az emberi objektumnak

detektált részbe, de annak csak a jobb felső sarkát érinti. További példa feliratozott képekre a 0 Függelék részben található (0.1. ábra).

4.1.6 Vizualizáló komponens

Ahhoz, hogy a képi feliratpozícionálást hasznosítani lehessen szükséges egy olyan komponens, ami a megadott koordinátákra helyezi el az inputként szolgáltatott szöveget, ahogy ezt már az architektúrális ábrán is jeleztem (3.7. ábra). Rendelkezésünkre áll a feliratozandó kép, a méret és pozíciókoordináták, illetve a szöveg is. Ezek alapján szerettem volna minél esztétikusabb végeredményt elérni a felirat jó láthatósága és olvashatósága mellett.

4.9 ábra: Programozott feliratégetés eredménye

A prioritizáló komponens meghatározza a felirathoz tartozó szövegdoboz *x* és *y* koordinátáját és rendelkezésünkre áll a szövegdoboz hossza és magassága is. Zavaró lenne és sok információ letakarásával járna, ha ebben a méretben fedném el a vizuális tartalmat. Arra törekedtem, hogy a minimális információ kerüljön fedésbe, azaz csak annyi amennyi a szöveg elolvasását közvetlenül segíti. A bementi szöveg (*plain text*) tördelésén sok múlik, amely a prekoncepció alapján nincs semennyire sem tördelve, mert a későbbiek során az információ forrásának tekinthető *SRT* fájlban sem alkalmaznak ilyet. A szöveghossz alapján történő kettévágás jó megközelítés lehet csupán csak arra kell figyelni, hogy szóközök mentén történjen a bontás. Itt azzal a feltételezéssel éltem, hogy a bementei szöveget két sorban szeretnénk vizualizálni. Arial betűtípust használtam és a

betűméretet automatikusan a kép felbontásához igazítottam. Ezek alapján az alábbi képlettel határoztam meg a felirat végleges pozícióját, ahol az egyes változók *(x,y)* számpároknak felelnek meg (pl: objektum szélessége, magassága vagy koordináta vízszintes és függőleges komponense). A rajtuk végzett műveletek külön vonatkoznak a koordinátákra:

$$felirat\ pozíciója = szövedoboz\ pozíciója + szövegdoboz\ mérete - \frac{sorméret}{2}$$

A fehér felirat szövege alá elhelyeztem egy sötét téglalapot a könnyebb olvashatóság érdekében. Ennek az egyik megoldása látható a fenti ábrán (4.9 ábra), ahol az eredeti pozíció kitakarta volna a sárga autó *MUSTANG* feliratát. További példák a függelékben találhatók (ábra 0.2).

Tehát a pozíció meghatározása és a felirat elhelyezése is a megalkotott komponensek segítségével történt így elkészült a *képfeliratpozícionáló alrendszer*.

4.2 Videófeliratpozícionáló rendszer megvalósítás

Ahogy az a tervezési fázisban is olvasható (3.2 fejezet) a videófeliratpozícionáló algoritmus inputja videóból, bemeneti SRT fájlból és egy konfigurációból tevődik össze. Kimenete pedig egy darab ASS fájl, ami tartalmazza a feliratok módosított pozícióit. A videófeldolgozó komponens feladata ennek az SRT - ASS fájlkonverziónak és hozzá kapcsolódó pozíció információknak a megteremtése a bemutatott képfeldolgozó alrendszer segítségével. Jelen fejezetben a rendszer megvalósítása során előjött kérdéseket, problémákat és a rájuk adott megoldásokat ismertetem.

Az implementáció során 3 különböző algoritmust szerettem volna megvalósítani. Fix pozícionálást, mikor a felirat a helyzete az egész vizuális tartalom figyelembevételével keletkezik és egyetlen pozíció kiválasztása a cél. Stabil pozícionálást, mikor az egyes szegmensek helyzete stabil marad, de a feliratblokkok helyzete változik. És dinamikus pozícionálást, mikor a szegmensek pozíciója is dinamikusan igazodik a vizuális tartalomhoz. A komponens feladata a videó képkockákra bontása, majd továbbítása a képfeldolgozó komponens felé. Azonban a megfelelő képkockák kiválogatása a három módszernél különböző. Továbbá a képfeldolgozó kimenetén keletkező koordináták helyes transzformációja és kezelése is algoritmusonként változó lehet.

A módszerek különbözősége mellett szükség volt olyan általános megoldásokra, amelyek mind a három technika esetében jól használhatók. Ilyen például a fájlformátumok kezelését és szerializálását támogató osztályok. A bemeneti SRT fájlt fel kell olvasni és célszerű struktúraként reprezentálni a programkódban támogatva a könnyű kezelhetőséget. A reprezentáció szükségessége ugyanúgy igaz a kimeneti ASS fájlformátumra is ellenben itt a szerializálást szükséges megoldani. Mindkettő fájltípus emberi szemmel is olvasható, ezért pontosabb megértésük érdekében elhelyeztem egy-egy SRT és ASS fájl mintát (4.10 ábra és 4.11):

```
168
00:20:41,150 --> 00:20:45,109
- How did he do that?
- Made him an offer he couldn't refuse.
```

4.10 ábra: *SRT* fájl felépítése (részlet)

```
[Script Info]
; Script generated by Aegisub
; http://www.aegisub.org
Title: Neon Genesis Evangelion - Episode 26 (neutral Spanish)
Original Script: RoRo
Script Updated By: version 2.8.01
ScriptType: v4.00+
Collisions: Normal
PlayResY: 600
PlayDepth: 0
Timer: 100,0000
Video Aspect Ratio: 0
Video Zoom: 6
Video Position: 0

[V4+ Styles]
Format: Name, Fontname, Fontsize, PrimaryColour, SecondaryColour,
OutlineColour, BackColour, Bold, Italic, Underline, StrikeOut,
ScaleX, ScaleY, Spacing, Angle, BorderStyle, Outline, Shadow,
Alignment, MarginL, MarginR, MarginV, Encoding
Style: DefaultVCD,
Arial,28,&H00B4FCFC,&H00B4FCFC,&H00000008,&H80000008,-
1,0,0,0,100,100,0.00,0.00,1,1.00,2.00,2,30,30,30,0

[Events]
Format: Layer, Start, End, Style, Name, MarginL, MarginR, MarginV,
Effect, Text
Dialogue: 0,0:00:01.18,0:00:06.85,DefaultVCD,
NTP,0000,0000,0000,,{\pos(400,570)}Like an Angel with pity on
nobody\NThe second line in subtitle
```

4.11 ábra: ASS fájl példa

Az SRT fájlok esetében egy feliratszegmenshez csupán egy sorszám és egy időintervallum tartozik. Könnyen átlátható és egyszerű a fájlnak a felépítése, ezért számos programnyelven léteznek SRT kezelő források. Én a Python nyelvet részesítettem előnyben ezért a PySRT Library-t [35] használtam erre a célra. A dependencia használatát becsomagoltam egy olyan osztályba, ami csak szükséges metódusokhoz ad hozzáférést a számomra megfelelő formában.

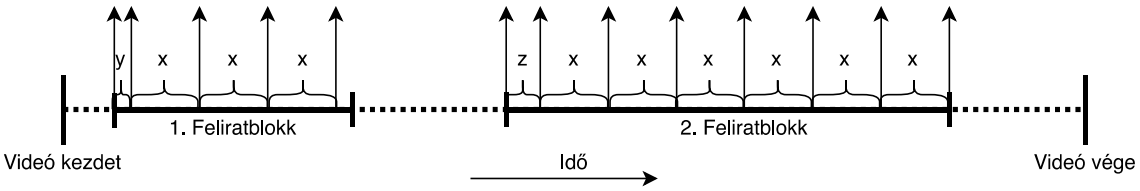
Az ASS fájlok kezelésére a PySubs2 könyvtárat [36] találtam a legmegfelelőbbnek. Az előző példa alapján látható (4.11 ábra), hogy ebben az esetben a feliratfájl formátum jóval bonyolultabb. Alapvetően három részből áll, egy metaadatot leíró "*Script Info"* részből egy stílusokat leíró „*V4+ Styles*" blokkból és a tényleges feliratokat tartozó „*Events*" részből. A példából látható, hogy a pozíciókoordináták leírása több szinten is megvalósítható. Egyfajta stílusként is meglehet határozni, ha arra van igény, de ettől függetlenül az *event*-eknél is van lehetőség a pontosításra. Mivel elsősorban külön-külön szerettem volna a blokkoknak új helyzetet adni, ezért az *event* pozíciókat módosítottam a későbbiek során.

Az, hogy az SRT-ASS transzformáció miként zajlik le milyen idő és pozíció attribútumok figyelembevételével, az már a pozícionáló algoritmuson múlik (ami az ismertettek alapján dinamikus, stabil és fix lehet).

4.2.1 Dinamikus pozícionálás

Dinamikus pozícionálás esetében arról van szó, hogy a legjobb feliratpozíciót jelentő koordináták folyamatosan (meghatározott időintervallumonként) kiszámításra kerülnek és a felirat ennek megfelelően mozdul el a videóban. Tehát az algoritmusnak szükség van egy időintervallumra, ami alapján a képfeldolgozó modul futtatásának sűrűségét meg tudja határozni. Az időintervallum értékét a rendszer konfigurációs bementi paraméterének tekintettem, így a későbbiek során könnyen tudtam különböző értékekkel számításokat végezni.

Legyen x az input időintervallum (másodpercben). Ekkor az algoritmus végig iterál a videó képkockáin és azokban az esetekben, mikor látható felirat az adott képkockán, lefuttatja a képfeldolgozó algoritmust az adott képkockára. Az alrendszer futtatásának az eredménye egy koordináta páros, amely a legjobb pozíciónak tekinthető az aktuális képkockán. Erről a mintavételezési módszerről készítettem az alábbi ábrát (4.12 ábra).

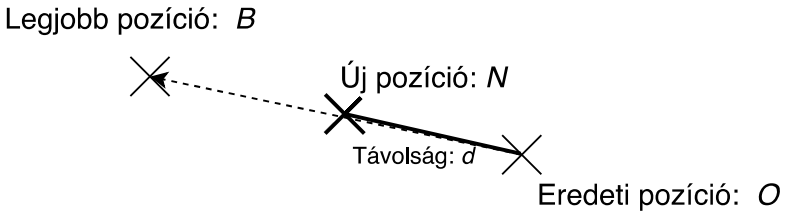


4.12 ábra: Dinamikus feliratpozícionáló mintavételezése

A vízszintes tengely egy videó hosszát reprezentálja, melyen a 2 feliratblokk található. Az algoritmus a videó elejétől elkezd iterálni a képkockákon és ha olyan kockához ér, minden *x*-edik időintervallumban, amelyikhez tartozik feliratszegmens is, akkor lefuttatja a képi feliratpozícionálót. Ezt jelölik a felfelé mutató nyilak az ábrán. Az algoritmus fejlesztése során láthatóvá vált, hogy nem elég csupán az *x*-edik időpillanatokat szemmel tartani mivel, ha a felirat épp nem az egy *x*-edik másodpercben jelenik meg akkor is valamilyen módszer alapján meg kell határozni pozícióját. Ekkor, ha az alapértelmezett alul-közép helyzetre helyezzük, a legközelebbi *x*-edik időpontban átugrásra kerül a számított legjobb pozícióba, amely felesleges feliratugrálást eredményez, ha ezzel az információval már az első megjelenésnél is tisztában vagyunk. Ezért a képfeldolgozó algoritmust akkor is le kell futtatni, amikor először jelenik meg egy felirat az adott képkocka sorozaton, így a legközelebbi pozícionálás rövidebb időintervallumba fog esni, mint *x.* Az ábrán (4.12 ábra) ez a *y* és *z* időintervallumoknak felel meg. Minden esetben $0 \leq y < x$ és $0 \leq z < x$ fennáll.

Dinamikus pozícionálás esetében nagyon fontos a generált pozíciók vizsgálata. Mivel minden *x*-edik időpillanatban érkezhet új pozíció, amely simán eltérhet az előzőtől, ezért, ha semmilyen műveletet nem végeznék a kiszámolt pozícióinkon minden *x*-edik időpillanatban új helyre kerülne a felirat. Feleslegesen zavarónak ítéltem meg azokat az eseteket, amikor egy feliratszegmensblokkon belül történik meg a nagymértékű pozícióváltás. Ezért szerettem volna egyfajta interpolációt elvégezni az egy feliratszegmenshez tartozó pozíció értékeken, hogy az egymásutáni pozíciók távolsága sokkal kisebb legyen. Meghatároztam egy konstans távolságértéket, legyen ez *d*, amely leírja, hogy két egymást követő pozíció egy feliratblokkon belül milyen távolságra lehet maximum egymástól. Mikor egy új legjobb pozíció kerül meghatározásra (B) a képfeldolgozó alrendszer által, akkor ezt az értéket egyfajta *irány*nak tekintem ($O \rightarrow B$) a jelenlegi helyzetre nézve (O) és maximum *d* távolsággal kerül elhelyezésre a felirat (*N)*. (Amennyiben $|O \rightarrow B| < d$, tehát a legjobb pozíció közelebb van az eredeti pozícióhoz,

mint a meghatározott távolság, az új pozíció egyben a legjobb pozíció is lesz: $N = B$). Az új pozíció számításának megértését segíti a következő ábra (4.13 ábra):



4.13 ábra: Feliratpozíció módosítása az új optimum irányában

A dinamikus pozícionálás algoritmus működését pszeudokódos formában összefoglaltam (4.14 ábra). A `frame` változó azonosítja a videó képkockáit. Számosságából ki lehet számolni a videó képkockájához tartozó időtartamot `frame_count / frame_per_second` (*fps*). A feliratkezelő objektumtól (`subtitle`) pedig lekérdezhető, hogy adott időpillanatban van-e látható feliratblokk. Ha új feliratblokkról van szó, vagy a mintavételezési idő eltelt (`time_triggered`), akkor kiszámításra kerül a legjobb pozíció, majd a régi pozíciót eltoljuk a kiszámított legjobb irányába (`move_position`) és így áll elő a feliratszegmens új pozíciója.

```
for frame in video:
  frame_time = frame_count / frame_per_second
  if subtitle.visible(frame_time):
    subtitle_block = subtitle.get_block(frame_time)
    if new_block(subtitle_block) or time_triggered(frame_time):
      best_position = porcess_image(frame)
      new_position = move_position(last_position, best_position)
      save_position(new_postition)
  frame_count++
```

4.14 ábra: dinamikus pozícionálás pszeudo algoritmusa

Ahogy látható az algoritmus működéséből, érdemes a mintavételezési időt (x) minél kisebb intervallumra állítani, hogy a legjobb pozíció felé történő mozgatás minél többször érvényesülni tudjon és a felirat pozíciója folyamatos javításra kerüljön. Ezzel együtt a mozgatás távolságát (d) is célszerű kicsi értékként definiálni, hogy a felhasználó számára inkább a mozgatás inkább egyfajta animációként hasson, mintsem ugrálásként.

Kipróbáltam többféle paraméterezési lehetőséget is, de legkényelmesebbnek a 0,1 másodperc mintavételi idő és 5 pixel távolság tűnt. Az értékek csak saját megfigyelésemen alapultak. Az $50\ \frac{pixel}{másodperc}$-es sebesség könnyen követhető volt a szememmel és nem ugrálásként hatott, hanem inkább mozgásnak, animációnak.

Összességében elmondható, hogy ezzel a módszerrel nagyon jól igazítható a felirat egy mozgó személyhez vagy objektumhoz. A következő ábrán a tesztvideóból látható 3 egymásutáni részlet balról jobbra (4.15 ábra). Megfigyelhető, hogy a személy beszéde alatt a kezeit behúzza a felirat pozíciójába, ezért a szegmens rögtön feljebb csúszik a nyakkendős területre, majd a karok lejjebb tartásásával újra lefelé mozog a szöveg.

4.15 ábra: felirat dinamikus igazodása a vizuális tartalomhoz

A dinamikus igazítás nagy előnye, hogy a feliratblokk mindig igazodni tud a legjobb pozícióhoz és együtt mozog a tartalom változásaival. Az előny egyben hátrány is felhasználói szemmel, mert a sok mozgás nehezíti a követhetőséget és sok esetben fárasztóbb, mint amennyi előnnyel jár. Ezért mindenképp szerettem volna egy olyan megoldást is kidolgozni, amely kevesebb dinamikát tartalmaz és szegmensen belül nem változik a felirat pozíciója. Erre a célra hoztam létre a stabil feliratpozícionáló algorirmust (4.2.2).

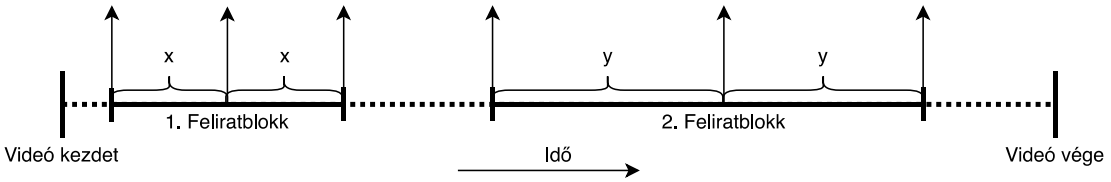
4.2.2 Stabil pozícionálás

A stabil pozícionáló algoritmussal szerettem volna megoldani a folyamatos mozgás problémáját, hogy az egyes feliratblokkok igazítása ne legyen zavaró a felhasználó szempontjából. A fő kérdés ezzel kapcsolatban az, hogyha az egyes feliratblokkok nem mozoghatnak jelenlétük alatt, akkor hova érdemes helyezni a feliratot,

ami összességében a legjobb pozíciónak felel meg a szegmensre nézve. A szöveg jelenléte alatt sokféle mozgás történhet a videón. Jelenetváltások is előfordulhatnak.

Eleinte jó megoldásnak tűnhet a dinamikus feliratpozícionálásnál ismertetett mintavételezés (4.12 ábra) használata majd a becsült pozíciók összegyűjtése és a középértékük meghatározása, mint új pozíció. Ezzel a megközelítéssel az a probléma, ha például a számított pozíciók a képernyő két szélére esnek, mert középen fontos vizuális elem látható, akkor eredményül pont a fontos tartalom fog kitakarásra kerülni a számított középérték miatt. Ezért egy eltérő megközelítést láttam célszerűnek.

A képfeldolgozó alrendszer működése során előállít úgynevezett *megfeleltetési map*-eket, amik leírják a fontos információkat a képeken. Ilyenre példa a már bemutatott autós ábra (4.7 ábra) is. Ezeknek a mapeknek egy mátrix felel meg melynek elemei a 0-1 tartomány értékeit veszik fel. A mátrixos megvalósítás miatt a *megfeleltetési map*-ek könnyen összeadhatók és normalizáció után ugyanúgy megjeleníthetők, mint az említett ábra (4.7 ábra). Az így kapott mátrix már nem csak egy, hanem kettő vagy akárhány kép *megfeleltetési map*-jeként fogható fel és optimális minimumhely található rajta. Tehát ha veszek egy feliratblokkot megkeresem hozzá a releváns képkockákat és egyesített *megfeleltetési map*-et gyártok hozzá a képfeldolgozó alrendszer segítségével, akkor az így kiszámított feliratpozíció az egész szegmens ideje alatt jól használható lesz. A szegmens nem fog zavaró pozícióba esni, mert az összegzett mátrix elemeiben ugyanúgy megjelennek az egyes képeken megtalálható fontos pixelek.

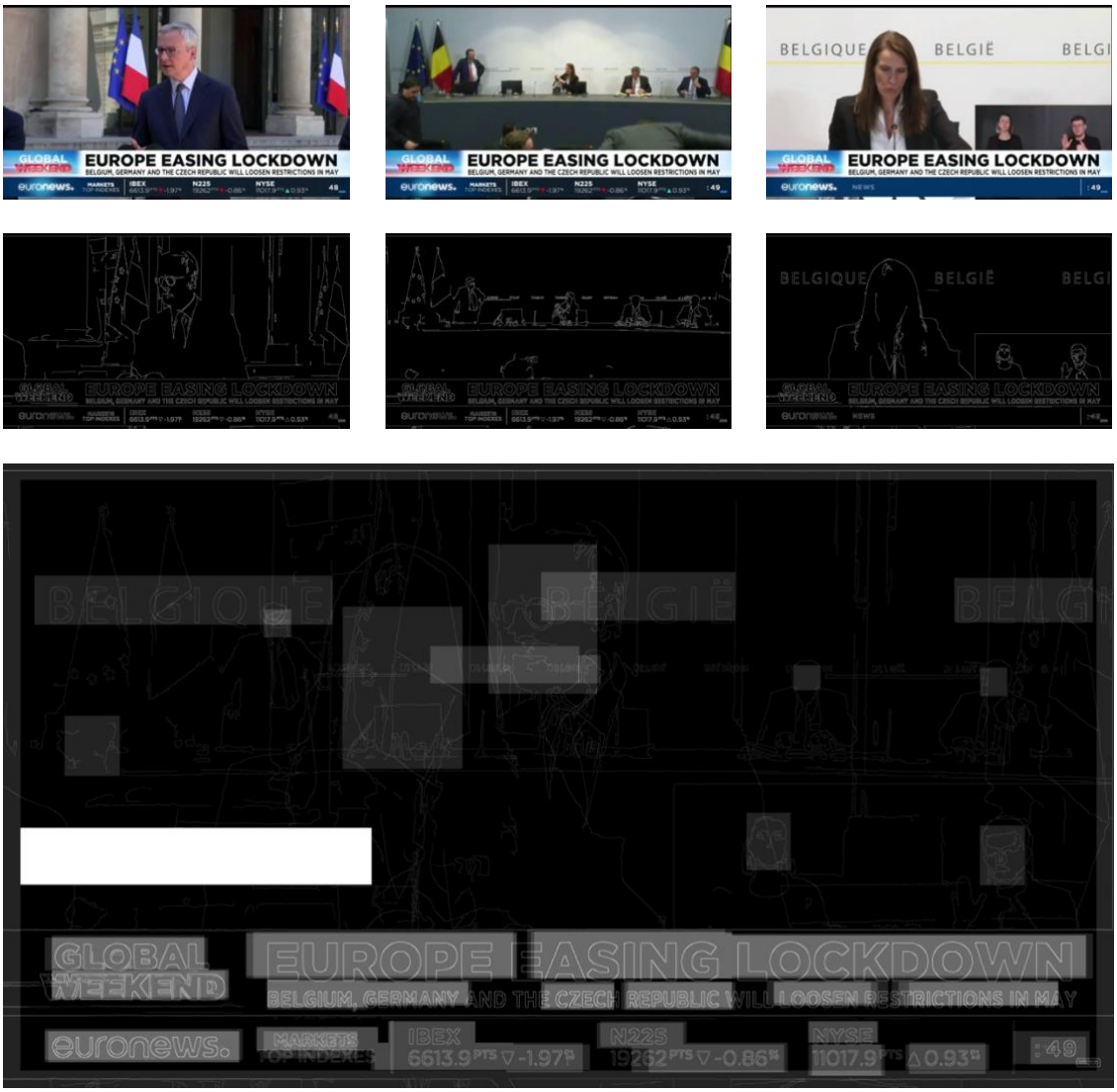


4.16 ábra: Stabil feliratpozícionálás mintavételezése

A releváns képkockákat úgy szerettem volna kiválasztani, hogy jól leírják a blokk ideje alatt látható vizuális tartalmat. Ezért az első képkocka mellé az utolsót is választottam, mivel sokszor előfordul, hogy a szegmens megjelenése alatt átlóg a következő jelentbe így igazodnia kell az ottani legjobb pozícióhoz is. A két képhez hozzávettem a szegmensidő közepén található képkockát is. Ezzel súlyozva azt a jelenet fontosságát a közös engedélyezési map-ekben, amelyik hosszabb időben szerepel a feliratblokk láthatósága alatt. Az algoritmushoz elkészítettem az előzőhöz hasonló

mintavételezési ábrát (4.16 ábra). Látható, hogy mindig három mintavételezés történik a szegmensek ideje alatt és nincs direkt időismétlődés a mintavételezésben ($x \neq y$).

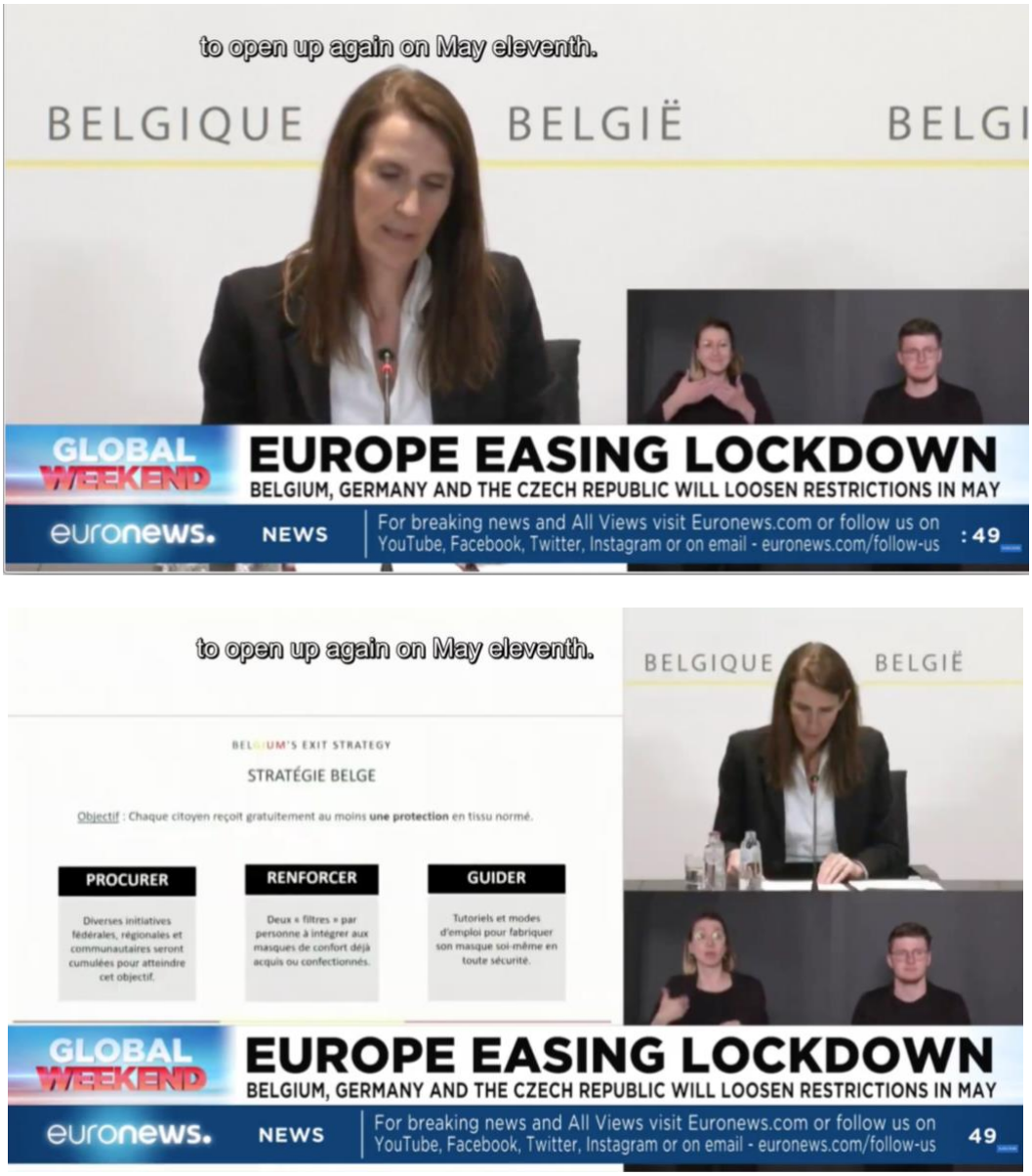
Az alábbiakban egy érdekes példa látható (4.17 ábra), mikor egy feliratblokk három különböző jeleneteben is látszódik. A legfelső sorban találhatóak az eredeti képkockák. Alattuk az éldetektált változataik. A legalsó sorban pedig az egységesített *megfeleltetési map*-jük. A közös ábrán megjelenik a különböző detektorok együttes eredménye is: a képen már nyolc arc is kiemelésre került szürke téglalapokkal, amely a detektorok találatát jelzi. A fehér téglalap az ajánlott feliratpozíciót mutatja a képek alatt megjelenő szegmensblokkra.



4.17 ábra: Stabil pozíció előállítása

A tesztvideókon kézzelfoghatóvá vált az algoritmus előnye. Erre láthatunk példát az alábbiakban (4.18 ábra). A két képkockán látható (a felső képkockát követte az alsó),

hogy a felirat pozíciója a kamerakép változást követően is megmaradt. Az algoritmus megtalálta mind a két jelenethez tartozó legjobb értéket. Ránézésre találhattunk volna jobb helyet a feliratnak az első képen (jobb szélen a *„BELGIË”* szó alatt), de az a második jelenetben már erősen zavaró lett volna.



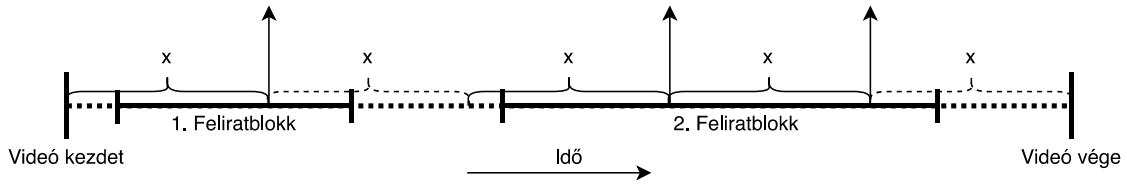
4.18 ábra: Stabil feliratozás példa

Tehát a stabil pozícionálással sikerült a mozgó feliratblokkok zavaró hatását megszűntetni és elérni, hogy egy szegmenshez egy, legjobb pozíciót találjon az algoritmus. Továbbá a számítás idő is drasztikusan lecsökkent, amelyre ebben az esetben nem a feliratozott időtartam van hatással, hanem a feliratblokkok száma. Azonban ezzel a technikával a különböző szegmensek különböző pozícióban jelennek meg továbbra is,

ami szintén zavaró lehet egyes nézők számára. Ennek a zavaró hatásnak a teljes megszüntetését szerettem volna elérni a fix pozícionálási technikával.

4.2.3 Fix pozícionálás

Fix pozícionálással szerettem volna a felirat mozgását teljesen megszüntetni. Kíváncsi voltatam, hogy kényelmi szempontból kedvezőbb-e, mint az eddig bemutatott változatok. Az előző két algoritmus (dinamikus és stabil) ötletei a fix pozícionálás során újra használtam. A videó mintavételezését ugyanúgy időintervallum alapján határoztam meg, mint dinamikus esetben, viszont a végleges pozíció számítása a stabil módszerhez hasonlított jobban. A *megfeleltetési map*-eket összegyűjtöttem és a hozzájuk tartozó mátrixok összeadásra kerültek, majd ezen az összesített ábrán történt meg a legjobb pozíció kiszámítása.



4.19 ábra: Fix feliratpozícionálás mintavételezése

A fix pozícionálási algoritmus mintavételezése a fenti ábrán látható (4.19 ábra). Az algoritmus input paramétere (*x)* leírja azt az időintervallumot, amilyen gyakorisággal a mintavételezés megtörténik, ha az időpillanatban felirat is látható a képkockán. Azért csak a feliratozott képkockák kerülnek kiválasztásra, mert nem lenne értelme olyan pillanatokat is figyelembe venni, amelyeken később nem jelenik meg pozícionált szöveg. A kiválasztásos mintavételezés követhető nyomon a videó időtengelyén (minden *x*-edik időpillanatban, mikor van felirat a képen megtörténik a mintavételezés). Az időalapú megközelítés előnye, hogy a hosszabb ideig látszódó szövegblokkok képkockái többször kerülnek kiválasztása, ha idejük eléri vagy meghaladja a mintavételezési idő kétszeresét. Ezáltal minél hosszabb időben egy szegmens annál súlyosabban reprezentálódnak a képkockái a közös *engedélyezési map*-ben. A fenti ábrán (4.19 ábra) ez is jól látható: a második feliratblokkhoz tartozó pozíció számítás kétszer is megtörténik mivel hosszabb szegmensről van szó. Így a videó közös engedélyező mátrixában a három mintavételezés detektor eredményei összesülnének, melyből kettő a második szegmenshez tartozik.

A megvalósítás során többfajta *x* értéket is kipróbáltam. Nem láttam célszerűnek 1 másodpercnél hosszabb időintervallummal számításokat végezni, hogy a rövidebb feliratok se maradjanak ki az összesítésből. 0,5 másodperces értékkel számolt végleges pozíció csak pár pixelben tért el az 1 másodperces esetnél, a számítási időt viszont jelentősen megnövelte. Ezért a későbbiekben alapbeállításként tekintettem az 1 másodperces mintavételezésre.

Az alábbiakban látható egy olyan ábra (4.20 ábra), amely az egyik tesztvideó engedélyező map-jét ábrázolja, amit a fix pozícionálási algoritmus gyártott futása során. Látható, hogy a videó alsó részén szinte folyamatos volt a szalagcím, ezért az ott található feliratok jól olvashatóak. A kép felső részén kicsit elmosódva, de megjelennek a videó során detektált arcok és a hosszabb ideig jelenlévő objektumok is. Érdekes ezt összehasonlítani a stabil pozícionálásnál bemutatott megfeleltetési map-el (4.17 ábra). A stabil algoritmus esetén megjelent detektálási eredmények a fix pozícionálás esetén is megjelennek, viszont a sok más képkockához tartozó detekciók mellett csak halványan láthatók. (Az ábrán nem jelöltem meg a végleges szöveg helyzetét).

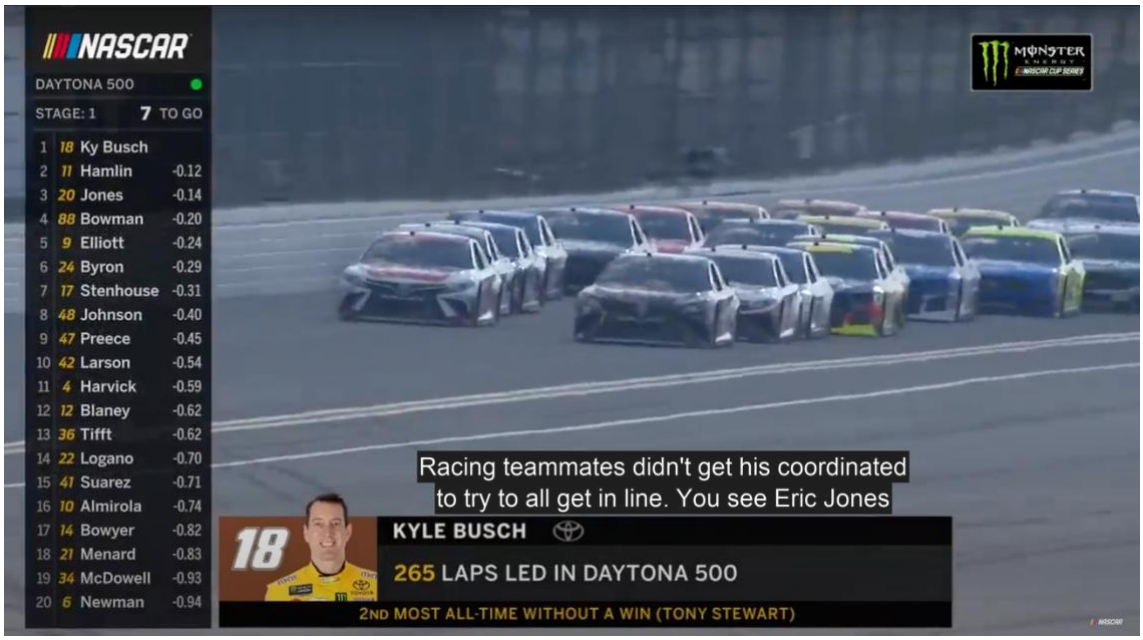
4.20 ábra: Tesztvideó engedélyező map-je fix pozícionálás esetén

5 Kiértékelés

Munkámnak ezen szakaszában szerettem volna összesíteni a megoldásom eredményeit és egyfajta jósági tényezővel meghatározni a rendszer képességeit. A korábbi felépítéshez hasonlóan jelen fejezetben is ketté osztottam a rendszer értékelését, először kép-, majd később videó feliratpozícionálási részhez tartozó fejezetekre.

5.1 Képfeliratpozícionáló kiértékelése

A képeket feliratozó alrendszert többfajta tényezővel lehet leírni. Vizsgálhatjuk a releváns objektumok, élek, képek, logók stb megtalálását, vagy a rendszer futási idejét is. Korábbi fejezetben (4.1 fejezet) már kitértem az egyes detektorok erősségeire és korlátaira. Szerettem volna felhasználhatósági szempontból is megvizsgálni és lehetőségekhez mérten javítani az alrendszer működését. A képfeliratozó jól hasznosítható kell, hogy legyen videós környezetben, ugyanis használatának ez lenne az elsődleges célja. Minél több detektort viszünk a rendszerbe annál több információt nyerünk ki a képekből. Azonban figyelni kell, hogy használatuk mennyi plusz számítási időt visz a működésbe a kinyerhető releváns információk arányában. A kiértékelésem célja azon detektorok megtalálása volt, amelyek videós környezetben is megállják a helyüket futási idő és releváns információ kinyerés tekintetében.



5.1 ábra: tesztkép

A detektorok futási ideje a detektálandó kép méretétől és nem a rajta található információktól függ, ezért nem láttam értelmét különböző képes kiértékelés elvégzésének. Azt szerettem volna vizualizálni, hogy egy detektálási algoritmus számítási időt tekintve hogyan teljesít a többi megoldáshoz képest egy adott képen. A választásom egy olyan képre esett, amelyen látható minden detektor számára releváns információ. Az egyszerűség kedvéért a kép sikeresen feliratozott változatát helyeztem el a fenti ábrán (5.1 ábra). Ezen vizsgáltam meg a futási időket. A detektorok az elvártak szerint megtalálták az éleket, szövegeket, autókat és személyeket, de a Monster illetve Nascar logók nem kerültek kiemelésre a logódetektor korlátozott felismerési képességei miatt. A futási idők az alábbiak szerint alakultak (3. táblázat):

Canny éldetektor	***0.68395 mp***
Szöveg detektor	***1.45674 mp***
Logó detektor	***9.18339 mp***
Objektum detektor	***20.93943 mp***
Arc detektor	***1.54057 mp***

3. táblázat: detektor futási idők

A táblázatból egyértelműen látszik, hogy az objektum és logó detektor használata jelentősen megnöveli a számítási időt. Képfeliratok pozícionálásánál kevésbé zavaró, ha 20 másodpercet kell várnunk a helyes pozíció meghatározására, de videós környezetben mikor képkockák százait kell feliratozni, nagyon megnehezíti a hosszú várakozás a hatékonyságot és az előrehaladást. Mivel a logó detektorhoz tartozó neurális hálózat csak kevés márkajelzés felismeréséhez alkalmas használatáról könnyű volt lemondani a videós környezetben. A kitalált rendszerarchitektúrának köszönhetően ez csak egy paraméter átállításával járt.

Az objektum detektor sok lényeges információt emel ki képeken és használata sokszor jó irányba befolyásolja a végeredményt. Ilyenre példa a 4.8. ábra feliratozása is. A közel 21 másodperces várakozás azonban elég erős negatívum. A probléma megoldásához azonosítottam a detektor legfontosabb használati esetét: az arcok detektálását. Nagyon fontos a precíz kiemelés ebben az esetben, mert kifejezetten zavaró, ha a felirat egy arcra kerül. Más objektumok felismerése az éldetektor által is megvalósul, mert körvonalaik kiemelésre kerülnek a helyes pozíció megtalálásánál. Ezért úgy

döntöttem, hogy az objektum detektor helyett arc detektálást fogok végezni, ami sokkal gyorsabban megvalósítható egy már meglévő forrás használatával [37]. A megoldás használata könnyen integrálható rendszerembe, hisz már volt detektor készítésben tapasztalatom és új algoritmus bevitele könnyen elvégezhető az architektúrának köszönhetően. Az arc detekció futási ideje már bemutatott táblázat alsó sorában is látható (3. táblázat).

Tehát a képfeldolgozó alrendszer kiértékelése után, videós környezetben az él-, a szöveg és az arc detektorokat használtam. Így egy képkockán történő pozícionálás sokkal gyorsabban elvégezhető volt anélkül, hogy lényeges információt hagytunk volna figyelmen kívül a legjobb pozíció meghatározásakor.

5.2 Videó feliratpozícionáló

Képek esetén a felirat helyének meghatározásakor az a legfontosabb, hogy az elhelyezendő szöveg ne takarjon ki fontos információt. Az, hogy a képnek melyik szegletére kerül a szegmens, kevésbé lényeges. Videós környezetben eltérnek a preferenciák. Rövid ideig el tudunk tekinteni attól, hogyha egy fontosabb objektumot takar ki a felirat, ha cserébe kevesebb ugrálást tapasztalunk a felirat pozíciójában. Az ugrálás hatásának csökkentésére és a jobb pozíció meghatározására dolgoztam ki a három pozícionáló algoritmust. ***Dinamikus***, mikor a felirat pozíciója teljesen dinamikusan igazodik a képi tartalomhoz. ***Stabil***, mikor az egymásutáni szegmensek helyzete változhat csak. És ***fix***, mikor már nincs mozgás egyáltalán a feliratban. Értelemszerűen, ahogy a dinamika csökken a módszerekben, úgy nő a kitakart információ lehetősége is.

Jelen fejezetben az összetett videó feliratozó rendszer értékelését végzem el néhány előre definiált jósági tényező alapján. Az egyik jósági tényező a számítás gyorsasága, mert fontos, hogy belátható időn belül készüljön el a pozícionálás. A pozíciók változásának jósága erőteljesen szubjektív tényező. Különböző emberek, különböző mértékben szeretik és viselik el a helyzetváltozást, ezért egy közvéleménykutatás keretében szerettem volna megvizsgálni, hogy melyik algoritmussal állítható elő általánosságban a legjobb eredmény.

Ahhoz, hogy jól látszódjon a pozíció változtatásának előnye megfelelő tesztvideót kell választani. Fontos, hogy a videó megfelelően rövid legyen, hogy a tesztalanyok ne veszítsék el érdeklődésüket a teszt kitöltése során. Előnyös, ha az eredeti felirat zavaró pozícióba esik, hogy jól látszódjon a megoldásom szükségessége. Illetve az is szerencsés,

ha a rövid videó alatt, minél több jelenet játszódik le, hogy a tesztelés során különböző helyzetekben lássunk példát az algoritmusok működésének eredményére. A szempontok figyelembevételével az Euronews híradó egyik részletét választottam ki tesztvideónak. 1 perc 8 másodperces részletről van szó, amelyben összesen 12 különböző tulajdonságokkal rendelkező jelenet, vágás van és végig hallható alatta a narrátor beszéde. Az eredeti feliratpozíció nagyon zavaróan a szalagcím helyzetében található. A tesztvideó részletei láthatóak különböző ábrákon (4.15, 4.17 és 4.18 ábra).

5.2.1 Futási idők összehasonlítása

Tehát a már bemutatott algoritmusok futási ideje az 1:08 perces videón az alábbiak szerint alakult (4. táblázat):

Dinamikus pozícionálás (0,1 mp-es mintavétel)	*4129.52 mp*
Stabil pozícionálás	*905.18 mp*
Fix pozícionálás (1 mp-es mintavétel)	*376.72 mp*
Fix pozícionálás (2 mp-es mintavétel)	*186.04 mp*

4. táblázat: pozícionáló algoritmusok futási ideje

A táblázatból egyértelműen kiolvasható, hogy minél kevesebb mozgással járó technikát használunk, annál gyorsabb számítási időt kapunk és így akár nagyságrenddel csökkenthetjük a számítás idejét. A fix pozícionálási technika mondható a legkedvezőbbnek számítási időt tekintve a közel 6 perces eredményével. Fontos kiemelni, hogy amennyiben a számítás gyorsabb elvégzése a cél lehetőségünk van más paraméter értékkel próbálkozni. Például a táblázat legutolsó sorában (4. táblázat) kétszeresére növeltem a mintavételezési időt, amely kétszeres gyorsulással is járt.

5.2.2 Eredmények összehasonlítása

Az említettek alapján szerettem volna az algoritmusok eredményét egy közvéleménykutatás keretében is összehasonlítani. A bemutatott tesztvideót négyféle feliratozott változatban készítettem el. Elhelyeztem rajta pozícionálás nélküli, dinamikus, stabil és fix módszerekkel generált feliratot a videóba égetve és feltöltöttem egy népszerű videómegosztó platformra [38][39][40][41]. A szöveget a videókon ugyanolyan betűtípussal és mérettel hoztam létre és az olvasását sem könnyítettem meg kontrasztos

háttér megjelenítésével. Szerettem volna, hogy csak a feliratok pozíciója legyen mérvadó a videók megítélése közben. A teszthez elkészítettem egy kérdőívet melyben ugyanazt a négy kérdést tettem fel a négy különbözően feliratozott videóról:

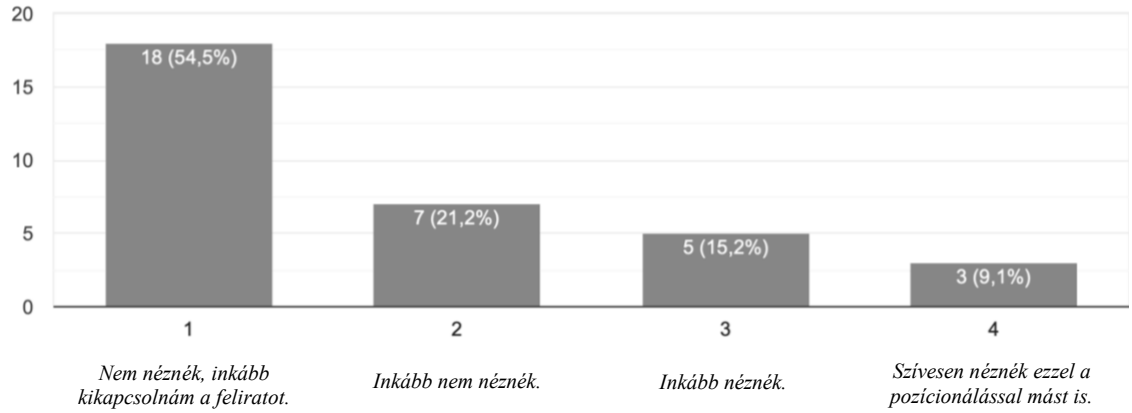
- *Mennyire találta kényelmesnek a felirat elhelyezkedését?*
- *Mennyire találta zavarónak a felirat mozgását/mozdulatlanságát?*
- *Mennyire szívesen nézne ezzel a felirat elhelyezésmóddal más videókat, filmeket?*
- *Összességében mennyire volt elégedett a videó feliratozásával?*

A válaszadási lehetőségeket 1-től 4-ig terjedő skálára korlátoztam mind a négy esetben. Szerettem volna elkerülni a középérték választásának lehetőségét, hogy a kitöltő egyértelműen döntsön az inkább pozitív, vagy inkább negatív élményéről. A 4-es számhoz társítottam a legpozitívabb élményt, az 1-eshez a legnegatívabbat. Ennek köszönhetően a válaszadások súlyozott összegéből egyértelmű rangsor állítható fel a videók között. A kérdőívet publikussá tettem ismerőseim számára [42].

A kérdőívvel nem volt célom reprezentatív statisztikai minta és kutatás felállítása, nem érdeklődtem a kitöltő személyek életkoráról, neméről és egyéb adataikról csupán a véleményük érdekelt, amivel szerettem volna alátámasztani saját megérzésemet és eredményeimet, hogy ne egyoldalúság vezéreljen az értékelés alatt. A kérdőívemet 33-an töltötték ki.

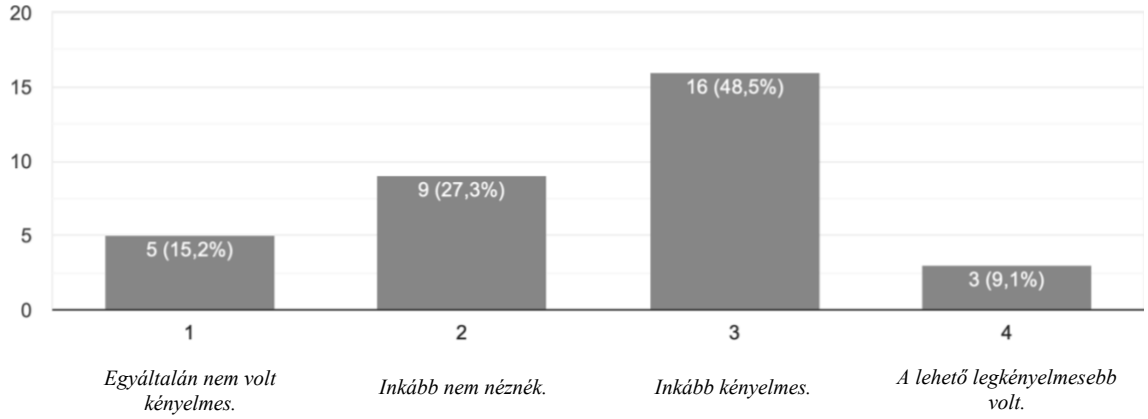
Az alábbiakban kiemelem a legérdekesebb eredményeket az egyes videók kapcsán grafikonnal ábrázolva a válaszok arányát, majd egy összesítéssel zárom a fejezetet.

Az első, nem pozícionált feliratozás esetében a legmeglepőbb válaszokat a „*Mennyire szívesen nézne ezzel a felirat elhelyezésmóddal más videókat, filmeket?"* kérdésre kaptam. Ugyanis a válaszadók több mint fele a legnegatívabb *„Nem néznék, inkább kikapcsolnám a feliratot"* választ jelölte meg (5.2 ábra, első oszlop), ami egyértelművé tette, hogy a feliratpozícionálás szükségessége felhasználói szemmel is érzékelhető ugyanis a legtöbb videót ezzel, a pozícionálás nélküli technikával nézzük.



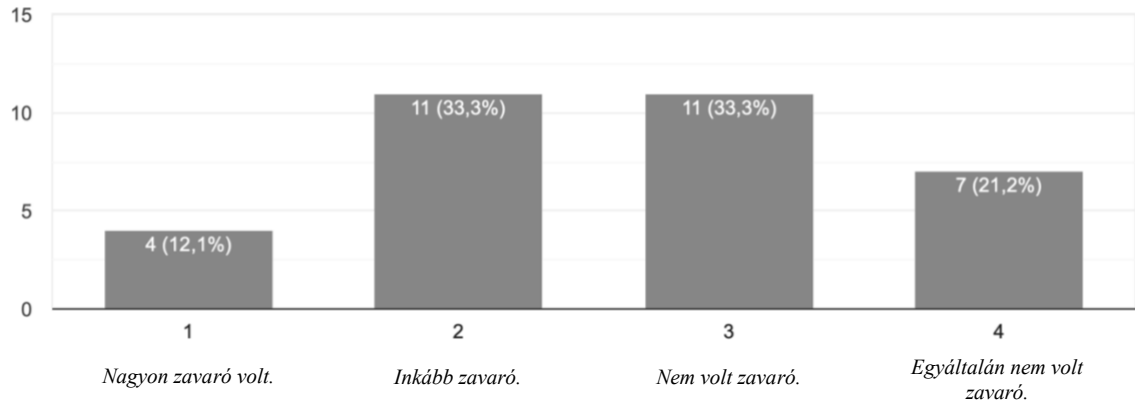
5.2 ábra: "Mennyire szívesen nézne ezzel a felirat elhelyezésmóddal más videókat, filmeket?" válaszadások a pozícionálás nélküli videón

Dinamikus elhelyezés esetén számomra a legérdekesebb válaszok a *„Mennyire találta kényelmesnek a felirat elhelyezkedését?”* kérdésre jöttek (5.3 ábra). A sok mozgás véleményem szerint nem volt kényelmes nézői szemmel, ezért magasnak tartom a közel 50%-os „*inkább kényelmes”* válaszarányt (3-ik oszlop), de az alacsony *„a lehető legkényelmesebb kényelmes volt”* (4-ik oszlop) arányából látható, hogy még el tudnak kényelmesebb verziót képzelni a kitöltők.



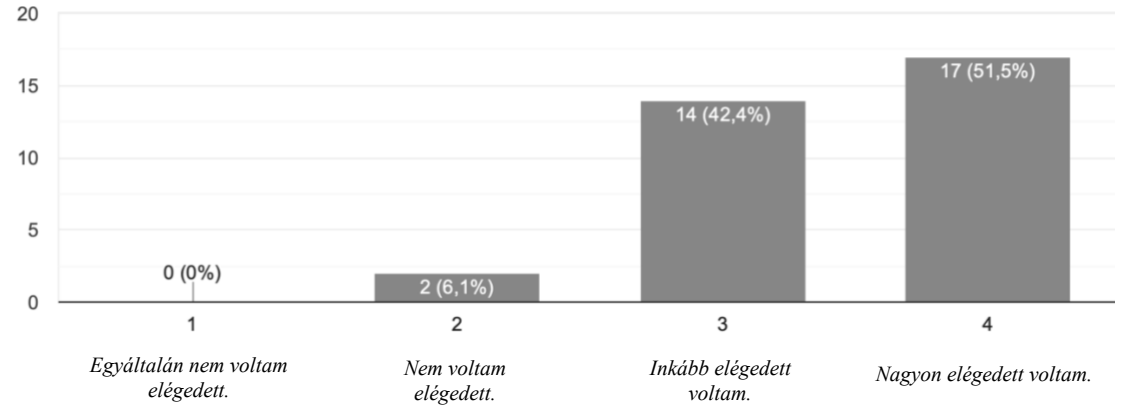
5.3 ábra: "Mennyire találta kényelmesnek a felirat elhelyezkedését? Válaszadások dinamikus pozícionálás esetén

Stabilizált pozícionáláshoz tartozó legérdekesebb kérdésként a *„Mennyire találta zavarónak a felirat mozgását/mozdulatlanságát?”* választottam. A válaszadók 33%-a tartotta „*inkább zavaró”* -nak (2-ik oszlop) a stabil pozíciókat, ahogy a „*nem volt zavaró”* (3-ik oszlop) választ is 33% esetben választották. Ebből látható, hogy a feliratblokkok mozgása nagyon megosztó nézői szemmel (5.4 ábra). Az, hogy a kevesebb mozgás népszerűbb, abból is látszódik, hogy ugyanerre a kérdésre dinamikus esetben 10 *„Nagyon zavaró volt”* (1. oszlop) és 14 *„Inkább zavaró”* (2-ik oszlop) válasz érkezett.



5.4 ábra: "Mennyire találta zavarónak a felirat mozgását/mozdulatlanságát?" válaszadások stabilizált pozícionálás esetén

Végül a fix pozícionáláshoz tartozó kérdések közül az *„Összességében mennyire volt elégedett a videó feliratozásával?”* válaszokat emelném ki (5.5 ábra). Ugyanis ebben az esetben sikerült elérni a legtöbb *„Nagyon elégedett voltam”* (4-ik oszlop) választ, ezzel egyértelműen láttatva, hogy az ismertetett technikák közül a fix pozícionálással jár a legmagasabb elégedettségi szint.



5.5 ábra: "Összességében mennyire volt elégedett a videó feliratozásával?" válaszadások fix pozícionálás esetén

A válaszok összesítéseként minden videóhoz és kérdéshez kiszámoltam egy jósági számot, ami a kérdésekre adott válaszok (1-4) átlaga. Az eredményeket az alábbi táblázatba gyűjtöttem össze (5. táblázat):

	Mennyire találta kényelmesnek a felirat elhelyezkedését?	***Mennyire találta zavarónak a felirat mozgását/ mozdulatlanság-át?***	***Mennyire szívesen nézne ezzel a felirat elhelyezésmód-dal más videókat, filmeket?***	***Összességében mennyire volt elégedett a videó feliratozásával?***	**Átlag**
Pozícionálás nélkül	1,24	2,85	1,79	1,36	***1,81***
Dinamikus	2,52	2,03	2,15	2,45	***2,29***
Stabil	2,85	2,64	2,61	2,79	***2,72***
Fix	3,33	3,52	3,52	3,45	***3,48***

5. táblázat: Válaszok súlyozott összege

Tehát például fix pozícionálás esetén a negyedik oszlop 3,45-os értéke az előző ábrán (5.5 ábra) látott értékek átlaga: $(2 * 2 + 14 * 3 + 17 * 4)/33 = 3{,}45$. A megkérdezettek száma 33 volt. Az utolsó *„Átlag”* oszlopban az adott technikára adott összes válasz átlagát számoltam ki.

A táblázat soraiból kiolvasható, hogy minden kérdésben a fix pozícionálás kapta a legtöbb pontot és érte el a legmagasabb átlagot, tehát ezt az algoritmust érezték a legkényelmesebbnek a kérdőív kitöltői. Érdekes, hogy sorról sorra nő a technikák pontszáma, amely megerősíti, hogy jó irányba haladtam az algoritmusok fejlesztésével. A pozícionálás nélküli feliratozás csak a „*Mennyire találta zavarónak a felirat mozgását/mozdulatlanságát?”* kérdésben előzte meg a dinamikus és stabil esetet. Ez alátámasztja, hogy minél kevesebb a mozgás annál nagyobb az élvezeti faktor a videó nézése alatt, de nem elhanyagolható, hogy a mozdulatlan felirat milyen pozícióban helyezkedik el a videó egészét nézve, mert az *„Átlag”* pontszáma így is nagyobb lett a dinamikus és stabil feliratnak.

A látott értékek alapján kijelenthető, hogy a bemutatott algoritmusok közül a *legjobb* mind számítási idő és használhatósági szempontból a ***fix pozícionálás.*** Emellett a dinamikus és stabil megoldás is élvezhetőbb eredményt nyújt az eredeti feliratozásnál.

6 Összegzés és kitekintés

Lezárásként egy rövid kitekintést adok a témakört illetően, megemlítem a továbbfejlesztési javaslataimat, valamint egy rövid összefoglalást adok munkámról.

6.1 Kitekintés és más megközelítések

Jelen fejezetben szeretnék vázlatos ismertetést adni a munkámhoz kapcsolódó, de mégis eltérő megközelítésekről, amellyel bemutatnám, hogy milyen más szempontok alapján lehet és érdemes feliratok pozícionálásán elgondolkozni.

- Munkámban elsősorban olyan esetekre kerestem a megoldást, amikor az alaphelyzetben megjelenő felirat nagyon zavaró olvashatósági és értelmezési szempontból. A felirat arrébb helyezésével szerettem volna a kitakarásra kerülő információ veszteségét minimalizálni. Nem volt célom a pozíció módosításával plusz információt teremteni. Érdemes lehet a pozícionálással információt teremteni a néző számára ezzel segítve a megértést. Például olyan videók esetén mikor párbeszéd látható szereplők között, pozícionálással kiemelhető az aktív beszélő. Ilyen megközelítésekről bővebben írtam a 2 Irodalomkutatás feliratelhelyezéshez című fejezetben, ahol ismertettem a kapcsolódó megoldási módszereket is.
- Egy másik tématerület, mikor nem helyileg szeretnénk a pozícionáláson javítani, hanem az időbeli pozíciók pontosításán tervezünk változást elérni. Nagyon zavaró filmvetítések közben, ha a felirat időben elcsúszik a hangoktól. Ez teljesen blokkolni tudja a megértést, ha a videó nyelvét nem értjük. Ezzel a megközelítéssel foglalkozik *Alberto Sabater* az *Automatic Subtitle Synchronization through Machine Learning* című cikkében [9], ahol gépi tanulással orvosolja a problémát.
- Amennyiben teljesen meg szeretnénk szüntetni a kitakarásból adódó információ veszteséget elgondolkodhatunk egy, eddigiektől teljesen eltérő megoldáson: a feliratnak külön képernyőterületet hozunk létre egyfajta statikus háttérrel. Gyakorlatilag arról van szó, hogy a videót egy képkeretbe ágyazzuk és a képkereten mindenféle mozgás nélkül jelenítjük meg a feliratokat. Ezzel a videó felbontása csökkenni tud, de a felirat sose takar ki semmilyen lényeges elemet.
- Egy másik érdekes megközelítés képi feliratozás témakörében, mikor a felirat tartalma gyakorlatilag fontosabbá vagy egyenértékűvé válik a vizuális tartalommal.

Képregények esetén nagyon fontos, hogy a felirat jól olvasható legyen, akár más tartalom megjelenésének rovására is. Az egyedi megjelenés a szövegbuborékos formájában már sokkal inkább a kreatív művészi alkotómunka kihívásait rejti magában, mint az algoritmizálható viselkedésmódot.

6.2 Továbbfejlesztési javaslatok

Munkám felhasználható képek és videók feliratozására egyaránt. Mindkét területen van lehetőség fejlesztésekre. A képfeldolgozó modul esetén a detektorok javítását és újak implementálását emelném ki, videófeldolgozás esetén pedig új algoritmusok kialakítását.

A használt detektorok segítségével lehetőségünk van megtalálni képeken éleket, szövegeket, arcokat, objektumokat és néhány logót is fel tudunk ismerni. Új detektorok alkalmazásával témaspecifikussá tehető a feliratozás, amely kedvező lehet a néző szempontjából. Például egy élővilággal kapcsolatos ismeretterjesztő videóban érdemes lehet az állatok detektálását előnyben részesíteni a szöveg, vagy az arc detekcióval szemben. De a meglévő detektorok gyorsítása is kényelmesebbé tenné a jelenlegi felhasználást.

Egy másik továbbfejlesztési javaslat keretében emelném ki, hogy a jelenlegi architektúrával könnyen megvalósítható olyan algoritmusok használata, amelyek leszűkítik a képeken, videókon elhelyezendő felirat pozícióját. Ha például azt szeretnénk elérni, hogy a videó feliratai a jobb alsó térnegyedbe essenek, akkor csak egy olyan „detektort” kell implementálni, amelyik ezt a térnegyedet részesíti előnyben a másik hárommal szemben. Így végeredményben a rendszer a jobb alsó részbe helyezné a feliratokat (amennyiben nem takar ki fontosabb részletet). Ezzel ekvivalens módon a kép/videó bármekkora részére szűkíthető vagy tolható a feliratok elhelyezése.

A videófeliratozó modul továbbfejlesztéseként kiemelném a precízebb pozíció összehangoló algoritmus alkalmazását. Jelenleg a *stabil* pozícionálási algoritmus esetén, nincsenek egymásra befolyással a szegmens pozíciók. Érdemes lenne a *dinamikus* algoritmushoz hasonlóan egy pozíció összehangoló algoritmusban elgondolkozni, ami csökkentené az újonnan megjelenő feliratblokkok távolságát.

A felirat olvasás élvezhetőségének növelése érdekében jó ötletnek tartom az *ASS* formátum adta lehetőségek széleskörűbb kihasználását. Egy ASS feliratfájlban

különböző stílusok definiálhatók, amelyek nagyban segíteni tudják a gyors olvasást nagyobb kontraszt arány, betűméret, más betűtípus, betűszín használatával. Komolyabb érdeklődés esetén biztos, hogy ennek is érdemes lenne jobban utána járni.

6.3 Összefoglalás

Dolgozatomban bemutattam a feliratozás fontosságát és a jelenlegi automatizált megoldásokat a témakör ismertetése mellett. Irodalomkutatást végeztem a témakörben és rámutattam a pozícionálási megközelítésekre, melyek párbeszédek könnyebb megértését teszik lehetővé. Célom ismertetése után egy rendszerarchitektúrát terveztem, amely lehetővé teszi feliratok okos pozícionálását vizuális tartalmak alapján képekre és videókra egyaránt. Az algoritmus figyelembe veszi a megjelenő logókat, a képen már szereplő szövegeket, a különböző objektumokat, emberi arcokat és képes az élek detektálására is. Ezek után először megterveztem majd bemutattam három különböző videó mintavételezési és pozíció kezelő algoritmust, melyek segítségével dinamikusan mozgathatjuk a feliratot a kép vizuális tartalmával, de lehetőség van stabilabb pozíciók kialakítására is a fix pozíció tartás mellett. A rendszer használatával elkerülhetjük, hogy az elhelyezendő szöveg zavaró helyzetbe kerüljön megadott képeken és videókon, így a feliratozást személyre és tartalomra szabottan is el lehet végezni. A folyamat ismertetése alatt számos példaképet és videóképkockákat mutattam be, melyeket a rendszer fejlesztése közben használtam. Az eredményeket értékeltem jósági tényezők, futási idő és használhatóság alapján. Valamint egy rövid közvéleménykutatás keretében megvizsgáltam más emberek pozícionálási preferenciáit. A képfeldolgozó alrendszer javítási lehetőségeit kihasználtam és a videó feliratozó rendszerbe már gyorsabb számítási módszerekkel alkalmaztam. Az értékelésben kiemeltem a legfontosabb detektálási módszereket az él-, szöveg és arc detekciót. Majd meghatároztam a vizsgált algoritmusok közül a legjobb mintavételezést is: a fix feliratpozícionálást. Kitekintésként megemlítettem más megközelítéseket a feliratozás témaköréből, amelyek érdekességként kapcsolódnak munkámhoz. Végül továbbfejlesztési javaslatokkal zártam a diplomatervemet.

Irodalomjegyzék

[1] Yongtao Hu, Jan Kautz, Yizhou Yu, Wenoing Wang, *Speaker-following Video Subtitles,* arXiv:1407.5145v1, 2014.07, https://www.researchgate.net/publication/264123179_Speaker-Following_Video_Subtitles, (2019.11.18)

[2] Shea Laverty: *What is Open Captioning?*, https://www.techwalla.com/articles/what-is-open-captioning, (2019.11.09)

[3] Washington University: *What is the difference between open and closed captioning?* https://www.washington.edu/doit/what-difference-between-open-and-closed-captioning, (2019.11.18)

[4] Google Cloud Speech-to-Text Service, https://cloud.google.com/speech-to-text, (2019.11.08)

[5] Amazon Transcribe Service, https://aws.amazon.com/transcribe/, (2019.11.08)

[6] IBM Watson Speech-to-Text Service, https://www.ibm.com/cloud/watson-speech-to-text, (2019. 11. 08)

[7] Simon James: *5 Good Open Source Speech Recognition/Speech-to-text Systems,* https://fosspost.org/lists/open-source-speech-recognition-speech-to-text, (2019.11.08)

[8] VLC Media Player, https://www.videolan.org/vlc/index.html, (2019.11.10)

[9] Alberto Sabater: *Automatic Subtitle Synchronization through Machine Learning,* https://machinelearnings.co/automatic-subtitle-synchronization-e188a9275617, (2020.05.01)

[10] Open Toegankelijke, https://www.opentoegankelijk.be/exhibitors, (2019.11.18)

[11] Wataru Akahori, Tatsunori Hirai, Shigeo Morishima: Dynamic Subtitle Placement Considering the Region of Interest and Speaker Location, Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, pp. 102-109, ISBN 978-989-758-227-1, https://pdfs.semanticscholar.org/2864/cfd949e5b787b1fd22c05b5bb6450197a72e.pdf (2019.11.06)

[12] Apostolidis, E. And Mezaris, V., Fast shot segmentation combining global and local visual descriptors, 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp 6583-6587

[13] Katti, H., Rajagopal, A. K., Kankanhalli, M., and Kalpathi, R., Online estimation of evolving human visual interest, ACM Transactions on Multimedia Computing, Communications and Applications (TOMM), 2014

[14] Monaci G.: Towards real-time audiovisual speaker localization, 2011

[15] Wikipedia: *Comparsion of optical character recognition software,* https://en.wikipedia.org/wiki/Comparison_of_optical_character_recognition_software, (2019.11.19)

[16] Nithiroj Tripatarasit: *Logo Detection Using PyTorch*, (2018.06), https://medium.com/diving-in-deep/logo-detection-using-pytorch-7897d4898211, (2019.11.19)

[17] Ivan's Software Engineering Blog: *Robust logo detection with OpenCV,* https://ai-facets.org/robust-logo-detection-with-opencv/, (2019.11.19)

[18] Francois from TalkWalker: *10 best Image Recognition tools*, https://www.talkwalker.com/blog/best-image-recognition-tools, (2019.11.19)

[19] Divyansh Dwivedi: *Face Detection for Beginners,* https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9, (2019.11.19)

[20] Wikipedia: *Edge Detection*, https://en.wikipedia.org/wiki/Edge_detection, (2019.11.19)

[21] Nika Tsankashvili: *Comparing Edge Detection Methods,* https://medium.com/@nikatsanka/comparing-edge-detection-methods-638a2919476e, (2019.12.11)

[22] IBM Watson Media, *Video Enrichment*, https://www.ibm.com/watson/media/video-enrichment, (2020.04.14)

[23] Matroska: *SSA/ASS Subtitles,* *https://matroska.org/technical/specs/subtitles/ssa.html,* (2020.04.20.)

[24] Sofiane Sahir: *Canny Edge Detection Step by Step in Python,* https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123, (2019.11.20)

[25] Rodrigo Verschae, Javier Ruiz-del-Solar: Object Detection: Current and Future Directions, 2015

[26] OpenCV for Python, https://pypi.org/project/opencv-python, (2020.05.16)

[27] *Pillow*, https://pypi.org/project/Pillow/, (2020.05.16.)

[28] Arthur Ouaknine: *Review of Deep Learning Algorithms for Object Detection,* 2018, https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852, (2019.11.23)

[29] YOLO: *YOLO: Real-Time Object Detection,* https://pjreddie.com/darknet/yolo/, (2019.11.23)

[30] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiram He, Jiajun Liang: *EAST: An Efficient and Accurate Scene Text Detextor,* arXiv:1704.03155, 2017.04.11, https://arxiv.org/abs/1704.03155, (2019.11.24)

[31] Adrian Rosebrock: *OpenCV Text Detection (EAST text detector),* 2018.08.20, https://www.pyimagesearch.com/2018/08/20/opencv-text-detection-east-text-detector/, (2019.11.24)

[32] Ankur Singh: *Logo detection in Images using SSD*, 2018.07.11, https://towardsdatascience.com/logo-detection-in-images-using-ssd-bcd3732e1776, (2019.11.26)

[33] Akarshzingade, *Logo-Detection-YOLOv2 GitHub Repository*: https://github.com/akarshzingade/Logo-Detection-YOLOv2, (2019.11.26)

[34] *YOLOv2 logódetekcióval felismerhető márkák*: https://github.com/akarshzingade/Logo-Detection-YOLOv2/blob/master/obj.names, (2019.11.26)

[35] *PySRT* library: https://pypi.org/project/pysrt/ (2020.04.28)

[36] PySubs2: *https://pysubs2.readthedocs.io/en/latest/* (2020.04.28)

[37] Thanh Nguyen: *YOLOFace; Deep learning based Face detection using the YOLOv3 algorithm*, https://github.com/sthanhng/yoloface, (2020.05.02)

[38] Benda Krisztián: *Euronews without positioned subtitle,* https://youtu.be/iwpG50F_Eg4, (2020.05.03)

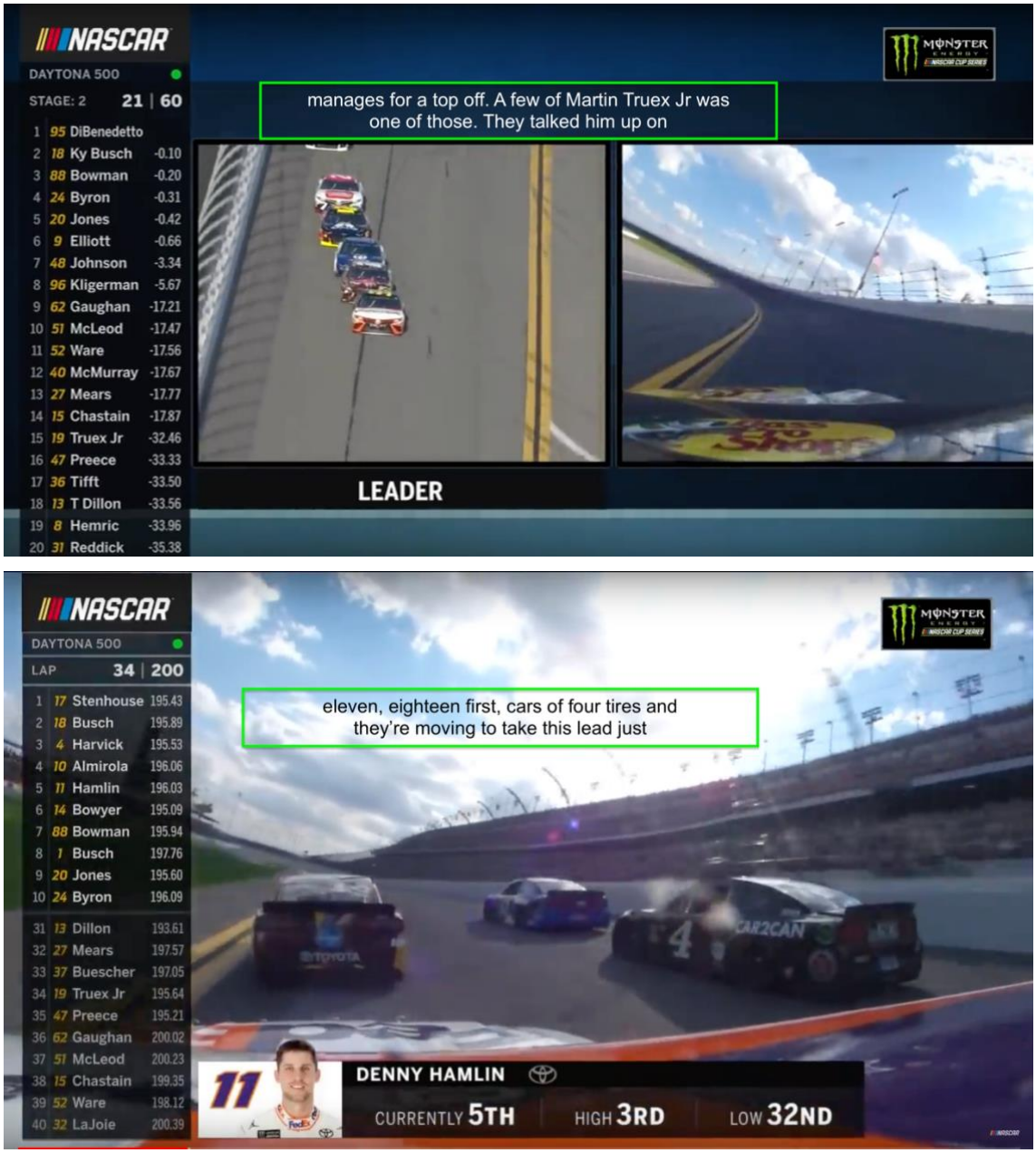
[39] Benda Krisztián: *Euronews with DYNAMIC subtitle positions*, https://youtu.be/RdfLZFXYrmQ, (2020.05.03)

[40] Benda Krisztián: *Euronews with STABILIZED subtitle positions*, https://youtu.be/RdfLZFXYrmQ, (2020.05.03)

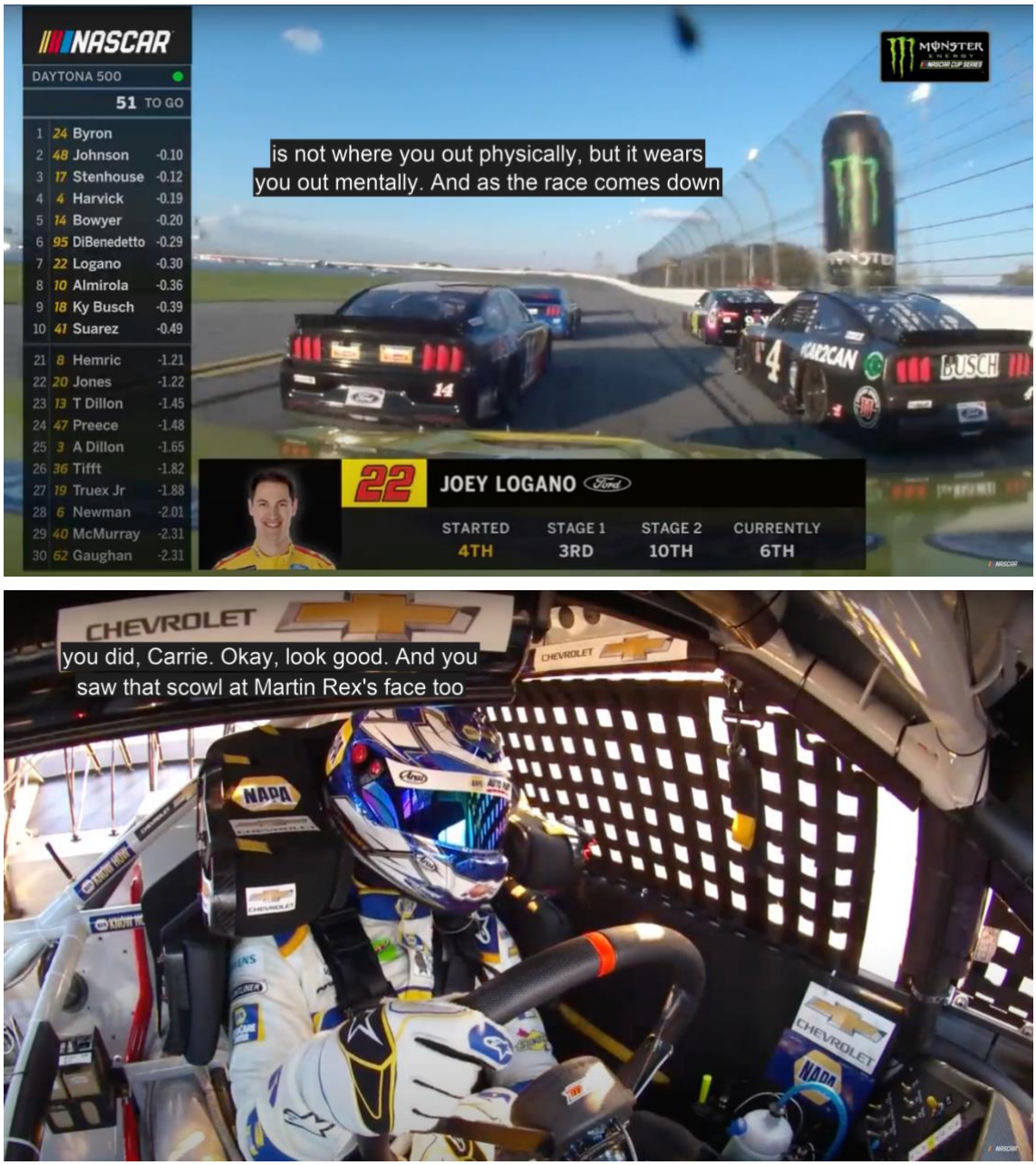
[41] Benda Krisztián: Euronews with FIX subtitle position, https://youtu.be/TZdb6gjCxzg, (2020.05.03)

[42] Benda Krisztián: Okos feliratpozícionálás kérdőív, https://forms.gle/PuJV1gCy8TWLcBCp9, (2020.05.03)

Függelék



0.1. ábra: Feliratozott képek egy Nascar videóból



ábra 0.2: A képfeliratpozícionáló rendszerrel elhelyezett és képre égetett feliratok