**Dokumentáció a Gépi Látási beadandóra**

*Vörös Bence László*

*C3EC1Z*

**Tartalomjegyzék**

[Bevezetés 2](#_Toc88817560)

[A megoldáshoz szükséges elméleti háttér 2](#_Toc88817561)

[A megvalósítás terve és kivitelezése 3](#_Toc88817562)

[Tesztelés 5](#_Toc88817563)

[Felhasználói leírás 7](#_Toc88817564)

# Bevezetés

Az általam választott és ki dolgozott téma a felhasználók által kiválasztott és bevitt fényképen szereplő rendszám felismerése, detektálása és le olvasása volt. A feldolgozott probléma az élet számos pontján applikálható, legyen szó akár egy autópálya használati jogosultságot ellenőrző rendszerről, adott autót vagy az azt használó személy mozgását meghatározó rendszerről melyhez akár felhasználhatja a térfigyelő kamerákat, a traffipaxok képei alapján a gyorshajtó autók detektálásáról, de akár csak egy egyszerű beléptető rendszerről mely a parkolni kívánó járművek rendszáma alapján meghatározhatja a fizetendő összeget vagy a használatra való jogosultságot is.

Mivel a képek forrása, minősége és a rendszámtáblához viszonyított nézőpontja és dőlése ismeretlen, így ezt a szoftvernek is képes kell lennie lekezelni és a helyes megoldást megtalálni (szinte) bármilyen bevitel esetén.

# A megoldáshoz szükséges elméleti háttér

A feladat megoldásához több elméleti algoritmust használtam melyek megtalálhatók beépítve az openCV Python könyvtárába.

A Canny egy olyan algoritmus amelynek célja az adott képen automatikusan történő, élek detektálása. Jelen esetben az élt mint a kép intenzitásában hirtelen, nagy mértékű megváltozás definiáljuk. A Canny lefutása 5 fő lépésből áll.

1. Zaj redukció: egy 5x5-ös kernelt használó Gauss filter,
2. Az intenzitás gradiensek megtalálása: egy Sobel kernel használatával mind horizontális, mind vertikális irányban egy filterezést végzünk. Az igy megkapott deriváltakkal majd meghatározzuk az él gradienseket. Ennek képlete

# A megvalósítás terve és kivitelezése

A rendszámtábla leolvasásához be kell látnunk, hogy az első és egyben legfontosabb (mindamellett legnehezebb) lépésnek, annak elhelyezkedésének pontos meghatározása kell, hogy legyen. Az egész képen történő karakter keresés hozhat sikeres eredményeket ám nem nehéz olyan példát találni, ahol már az autón vagy annak környezetén olyan alakzatok, karakterek találhatók melyek az ilyen módú leolvasást megnehezítik vagy lehetetlenné teszik.

Éppen ezért a rendszám helyének meghatározását kell célba vennünk előszőr. Itt számos megközelítés létezik, található olyan mely a rendszám hátterének színére (fehér) hagyatkozik ám ez semmiképp sem konstans, kifejezetten a zöld rendszámmal rendelkező elektromos hajtású autók elterjedése miatt, de akár csak a tábla tisztasága is befolyásolhatja a detektálást. Mivel az autó és a kamera térbeli relációja ismeretlen, nem is beszélve a rendszámtáblának az autón történő elhelyezkedéséről így erre sem hagyatkozhatunk. Az általam legoptimálisabbnak és a mások által legelfogadottabbnak tűnő megoldás az élek detektálására majd az azok által leírt poligonok szögeinek méretére, számára és oldalaik hosszának arányára hagyatkozik.

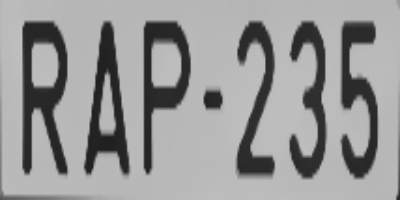
A program kivitelezéséhez én a Python nyelvet választottam, részben a fejlesztési folyamat gyorsasága és egyszerűsége miatt, de nem utolsó sorban a kiterjedt és széleskörű kód könyvtár potenciálja miatt. A képek beolvasására, tárolására, elő-, utó-feldolgozására az openCV könyvtárat csomagot választottam mely a hobbi gépi látásos körökben épp annyira elterjedt és sikeresen felhasznált, mint az iparban. A rendszámtábla karaktereinek leolvasására pedig a Tesseract OCR nevű szoftver mellett döntöttem, melynek a Python nyelvű implementációja a pytesseract könyvtár.

**Az algoritmus 5 főbb lépései:**

* A kép beolvasása és szürkeárnyalatossá alakítása
* Az élek és kontúrok detektálása
* Az ígéretesnek tűnő alakzatok egymás utáni feldolgozása amíg a helyes megoldást meg nem találtuk vagy az alakzatok el nem fogytak
* Az éppen tesztelt kontúr alapján az eredti kép egy szeletének kivágása és felkészítése leolvasásra
* Az előfeldolgozott képeken a karakterek felismerése

Ennek egy potenciális lefutását a következő képen mutatnám be mely egy optimális esetnek tekinthető. Első lépésként ezt át alakítjuk szürkeárnyalatossá.

Majd végre hajtjuk az éldetektálást. Ez Canny módszerrel történik 75-ös alsó és 250-es felső határértékekkel. A kontúrok felismeréséhez az openCV beépített RETR\_TREE és CHAIN\_APPROX\_SIMPLE konstansait használtam.

Majd a detektált kontúrokat területük szerint csökkenő sorrendbe rendezzük és így haladunk rajtuk végig egészen addig amíg a helyes választ meg nem találtuk vagy a vizsgált alakzatok területe a le nem csökkent a legnagyobb terület 90%-a alá, ez a paraméter viszont szabadon módosítható, szimplán a program lefutásának gyorsításáért felel. Az éppen feldolgozott kontúr a képen pirossal látható, az őt bekerítő négyzet pedig kékkel került megrajzolásra. Mellette látható a kontúr alapján kivágott képmetszet mely a rendszámtáblát tartalmazza.

Ezt követően futhat le a karakter felismerő algoritmus, a Tesseract. Ennek paraméterezése jelen esetben „--oem 1 --psm 11”, ahol az oem tag az LSTM neurális háló alapú szövegfelismerés kiválasztásáért felel, míg a psm-el a keresett szöveg szegmentálásának módját lehet leírni, jelen estben a 11-es egy véletlenszerűnek tekinthető elhelyezkedést határoz meg és így minél több karaktert próbál megtalálni. Ezután a leolvasott szöveget az indításkor megadott paramétereknek megfélően lehetőség van szűrni és amennyiben a kapott szöveg egyezik az elvárttal úgy vége a lefutásnak, ám ha nem akkor a következő alakzat tesztje következik.

Itt kitérnék a fenti leírás során futólag említett behatároló négyzet szerepére. Annak érdekében, hogy a feldolgozott alakzatokat lehetőség legyen valamely módon automatikusan megszűrni úgy, hogy csak azokat teszteljük melyek méretarányai hasonlóak egy potenciális rendszámtábláéhoz bevezetésre került egy ratioCheck nevű funkció melynek feladata ennek megsaccolása a széllesség és magasság adatok alapján. Ennek implementálása azért is volt fontos mivel, a program könnyen vakvágányra futhatott, ha elsőként egy olyan nagy területű zárt alakzatot vizsgált mely egyáltalán nem, vagy csak részben tartalmazza a keresett rendszámot. Itt, ha a hibásan detektált terület mérete eléggé nagy akkor lehetséges, hogy később hiába tesztelnénk a helyesen detektált rendszámtábla kontúrját a program a területi eltérés mértéke miatt nem venné figyelembe. Ezért, ha a vizsgált kontúr megfelelt a ratioCheck elvárásainak és területe nem szembetűnőén kisebb az eddigi maximum értéktől (ez alap esetben 70%ra van beállítva ám ez módosítható), úgy a megengedett maximumot módosítjuk az adott alakzatéra.

Illetve a program egy másik „különlegessége”, hogy amennyiben végigfutottunk már az összes potenciális alakzaton és már kifogytunk az opciókból, mint egy utolsó próba a detektálási algoritmus újra indul más kép előfeldolgozási lépésekkel, melyek bizonyos esetekben a megoldás helyes meghatározását eredményezik ám nem applikálhatok minden kép esetén ezért csak, mint egy utolsó opció kerülnek próbára.

# Tesztelés

A program működésének teszteléséhez egy 27 képből álló képsorozatot használtam, amelyek különböző autókat, különböző szögből, távolságból és viszonyok között ábrázolnak, ezeket az internen elérhető képek közű válogattam össze oly módon, hogy a lehető legjobban le teszteljem a szoftver határait. Ezek a képek a GitHub oldal .\images\ mappájában találhatok.

A képsor tesztelése során a program paraméter megadása nélkül 11 rendszámot volt képes sikeresen megtalálni, ez a -M paraméterrel 14-re, -R paraméterrel 13-ra, -M és -R paraméterrel 27 képből 16-ot ismert fel sikeresen.

Ennek számos oka volt, melyek a különböző forrású képek között változtak, mint például: a beviteli kép minősége, a kamera térbeli viszonya a táblához képest, a változó fényviszonyok, ám bizonyos esetekben az autó tartózkodási helye vagy típusa is problémát okozott a detektálás során. következőkben a tesztelt képek közül néhány szélsőséges esetet mutatnák be.

Mivel a rendszámtábla detektálása zárt, téglalap alakú poligonok észlelésére alapul ezért a következő képek feldolgozása során a rendszámtábla részleges takarása problémát okoz, de egy kifejezetten szögletes forma vagy világitás minősége könnyen tévútra viheti a detektálást.

# 

# Felhasználói leírás

A program fútatásához mindenképpen szükségünk lesz a python 3 telepítésére (fejlesztés során én 3.10et használtam). Továbbá szükség lesz az openCv és pytesseract könyvtárak telepitésére. Illetve szükség van a Tesseract OCR telepítésére is, fontos, hogy a telepítés után a kiválasztót mappa elérési útvonalát be kell másolnunk a kódba futtatás előtt (kivéve, ha az alapból felajánlott elérési útvonalat válasszuk, ami alap esetben C:\Program files\Tessarect OCR\).

A program futtatásához nélkülözhetetlen a -i parancssori paraméter megadása mely után egy elérési útvonalat kell megadnunk. Ez lehet egy fájl vagy egy mappa is (ez egy speciális eset amire a később térek ki).

A program rendelkezik továbbá 6 opcionális paraméterrel is:

* -B: ebben az esetben adhatunk csak meg egy mappát az i paraméter számára. Ha a program ezt az argumentumot észleli, bekapcsol egy úgynevezett „batch” módot, ahol a mappában szereplő összes képfájlra elvégzi a detektálási algoritmus futtatását. Majd ennek végeztével az utolsó sorban ki írja a sikeres találatok arányát.
* -V és -v: ez két különálló paraméternek is tekinthető ám mivel a feladatuk szinte megegyezik ezért egyben kezelem őket. A céljuk a „verbose” kimenet bekapcsolása, az-az az adott lépések közötti képek megjelenítése a felhasználó számára. A különbség a két megoldás között abban rejlik, hogy míg a -v minden lépés után egyesével jeleníti meg a képeket, a -V pedig egyben.
* -O: a „verbose” mód által megjelenített képeket el is menti (abba a mappába, ahonnan a program fútatásra került).
* -S: a „silent” mód bekapcsolására szolgál. Ez a gyakorlatban annyit tesz, hogy a program futása során semmilyen képet nem jelent meg és csak a szöveges kimenetet használja a felhasználó tájékoztatására.
* -h: a help üzenet megjelenítésére szolgál. Ez az egy eset, ahol a -i paraméter elhagyható. Továbbá amennyiben valamely másik paraméter hibásan lett megadva vagy a program más hibába ütközött úgy a program szinten ezt az üzenetet jeleníti meg.
* -M: megadásának segítségével a szoftver a képről leolvasott szövegre egy extra szűrési lépést is elvégez mely a kapott szövegben a magyar rendszám formátumnak megfelelő részletet keres. Ez hasznos lehet olyan esetekben, ahol a szövegleolvasás a rendszámon kívül más karaktereket is beolvasott (vagy valamely más objektumot detektált félre, mint karakter) vagy olyan esetekben, ahol a rendszámtáblán más karakter is szerepel (mint például az országot jelölő „H”). Fontos, hogy ez a szövegen nem módosít, karaktereket nem szúr be vagy töröl, csupán a kiolvasott karakterhalmaz egy részletét emeli ki ezzel javítva a detektálási arányon.
* -R: ezen opció megadásával a program amennyiben egy kontúr detektálása sikertelen volt, úgy kikalkulálja a rendszámtábla vízszinteshez viszonyított elforgatási szögét majd elforgatja a kivágott kép szeletet is és újra megkísérli a rendszám le olvasását ezzel javítva az eredményeket.