Multidiszciplináris tudományok, 10. kötet. (2020) 2 sz. pp. 303-318 **https://doi.org/10.35925/j.multi.2020.2.35**

RENDSZÁMTÁBLA AZONOSÍTÓ RENDSZER TERVEZÉSE PYTHON ÉS OPENCV HASZNÁLATÁVAL

Mosony Ákos
hallgató, Miskolci Egyetem, Informatikai Intézet, Alkalmazott Informatikai Intézeti Tanszék
3515 Miskolc, Miskolc-Egyetemváros, e-mail: mosonyakos@gmail.com

Hornyák Olivér
egyetemi docens, Miskolci Egyetem, Informatikai Intézet, Alkalmazott Informatikai Intézeti Tanszék
3515 Miskolc, Miskolc-Egyetemváros, e-mail: ohornyak@gmail.com

Absztrakt

A cikk egy rendszámtábla azonosító rendszer megvalósításának a felépítését ismerteti, mely képes egy digitális képről leolvasni a tábla karaktereit. A szakirodalomban szereplő rendszámtábla detektáló, illetve a karakterek felismerésére szolgáló módszereket is összegyűjti, valamint a Magyarországon forgalomban lévő rendszámtáblákra vonatkozó szabályokról is olvashatunk benne. Betekintést nyerhetünk a számítógépes látás világába, továbbá megérthetjük, hogy milyen képmanipulációs eljárásokon kell keresztül esnie egy digitális képnek, hogy arról kinyerhetők legyenek a karakterek. A rendszer Python nyelven íródott továbbá igénybe vette az OpenCV által nyújtotta lehetőségeket, valamint a Python-Tesseract optikai karakterfelismerő eszközt is a rendszámtábla karaktereinek azonosítására.

Kulcsszavak: *Rendszámtábla, Számítógépes látás, Optikai karakterfelismerés (OCR), Tesseract-OCR, OpenCV, Python*

Abstract

This article describes the structure of an implementation of a license plate identification system capable of reading the characters of a license plate from a digital image. It also collects the license plate detecting and character recognizing methods mentioned in the article, and we can read about rules regarding hungary's registered license plates as well. We can gain insight into the world of computer vision, as well as understand what image manipulation procedures a digital image must go through in order to extract characters from it. The system was written in Python and also used methods provided by OpenCV, as well as the Python-Tesseract optical character recognition tool to identify license plate characters.

Keywords: *License plate, Computer Vision, Optical Character Rrecognition (OCR), Tesseract-OCR, OpenCV, Python*

1. Bevezetés

A rohamosan fejlődő világunkban a járművek is szép számmal növekednek, ahol az azonosításuk számos esetben elengedhetetlen. Minden személygépjármű rendelkezik egy egyedi azonosítóval, amely gyakran egy alfanumerikus karaktersorozat, amit a forgalmi rendszámtábla jelenít meg. A rendszámtábla leolvasása a legegyszerűbb módja a járművek azonosítására, így meglehetősen nagy az igény a különféle rendszámtábla leolvasó rendszerekre. Napjainkban számos helyen alkalmaznak rendszámtábla felismerő rendszereket, csak gondoljunk a rendőrök által használt traffipaxokra, vagy a parkolóházakra, esetleg az okosotthonokra.

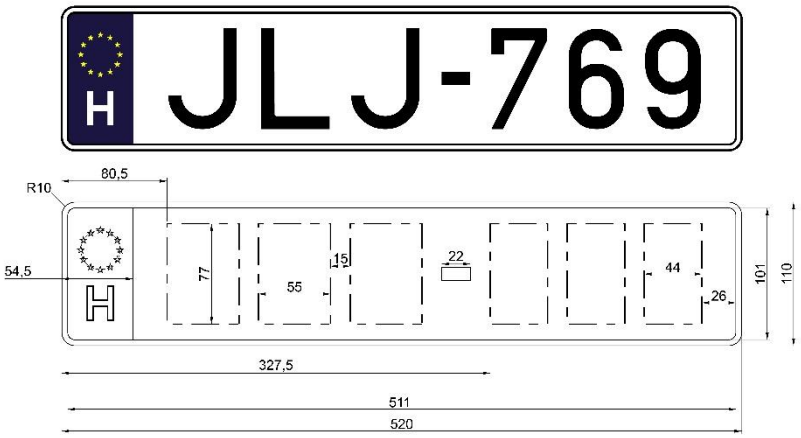
Amikor a rendszámtáblafelismerés témaköre felvetődik, akkor fontos tisztázni, hogy a rendszámtábla felismerés és a rendszámtábla azonosítás nem ugyanazt jelenti. Míg a rendszámtábla felismerésénél elég csak magát a táblát detektálni, addig az azonosításnál szükség van arra, hogy a karaktereket is felismerje a rendszerünk. A rendszámtábla azonosítás folyamatába egy fontos lépést valósít meg a maga a tábla detektálása, lokalizálása. Tehát rendszámtábla azonosítás egy olyan számítógépes látástechnika, amelylyel a járművek rendszáma kinyerhető egy digitális képből.

Jelenlegi áttekintő cikkben bemutatásra kerülnek azok a lépések és folyamatok, melyek szükségesek a rendszámtábla detektálásához, illetve a karaktereinek kiolvasására.

1.1. Forgalomba lévő rendszámtáblák Magyarországon

A 326/2011. (XII. 28.) Korm. rendelet 11. melléklete alapján, Magyarországon öt különböző rendszámtábla típust különböztetünk meg, melyek az „A”, „B”, „C”, „D”, illetve az „E” jelzést kapták. Az azonosításra szolgáló táblák hat darab alfanumerikus karaktert tartalmaznak, valamint az „E” típus kivételével elmondható, hogy három betű, és három szám kombinációjából tevődnek össze. A karakterek elrendezését tekintve csoportosíthatunk egysoros, kétsoros továbbá háromsoros rendszámtáblákat is, valamint ezek a csoportok méretükben is különböznek. A típusok és az elrendezés mellett fontos megemlíteni, hogy bérfuvarozók esetén sárga, illetve elektromos járművek esetén zöld az alapszín.

A cikk további részében az „A” típusú rendszámtábla kerül fókuszba, ahol a karakterek egy sorban helyezkednek el kötőjellel elválasztva, ahogyan az ***1.ábra*** is szemlélteti.



1. ábra. *„A” típusú rendszámtábla, és paraméterezése (a mérőszámok milliméterben értendők).*

Magyarország 2004. május 1-én csatlakozott az Európai Unióhoz, amely a forgalmi rendszámtáblánk tekintetében változást hozott. A csatlakozást követően baloldalt kék alapszínre rákerül az Európai Unió emblémája, a 12 sárga csillaga, valamint alá fehérrel az országjelzés.

Természetesen, mint sok más országban, hazánkban is lehetőség van egyedi rendszámtábla készítésére, melyekre a 326/2011. (XII. 28.) Korm. rendelet 58. § (1) bekezdésében leírtak vonatkoznak, továbbá a táblán lévő alfanumerikus karakterkészlet is megtalálható a kormányrendeletben.

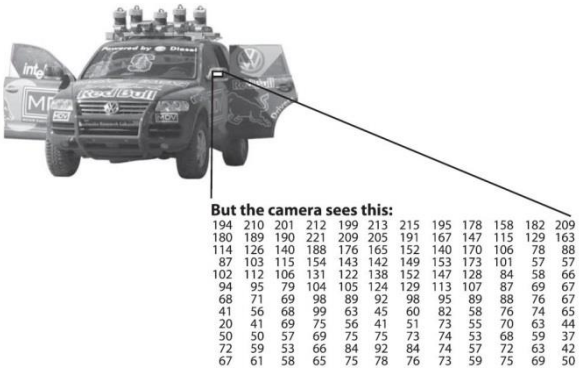
1.2. Számítógépes látás

A számítógépes látás egy rendkívül gyorsan fejlődő terület, és ezt többek közt a hatékonyságának köszönheti. Hasonlóan dolgozza fel a kamerából kinyert képeket, mint az emberi látórendszer, ahol az

agyunk a feldolgozó egységnek, a szemünk pedig a kamerának felel meg, azon belül pedig a helyes megvilágításért az íriszünk a felelős. Ez egy lehetőségekben gazdag, és kifizetődő kutatási terület az informatikusoknak, mérnököknek, és másoknak is, illetve egyre növekvő kereskedelmi jövője van. Manapság már az ipari felhasználásban is egyre elterjedtebb a számítógépes látás témaköre, csak gondoljunk arra, hogy kamerákkal ellenőrzik az alkatrészek méreteit, az ételek minőségét, de a csillagászatban vagy éppen az orvostudományban is előszeretettel alkalmazzák.[1].

Tehát a számítógépes látás egy digitális képen vagy videók esetén képeken történő adatátalakítás, és minden ilyen átalakítást valamilyen cél elérése érdekében végzik.

Mivel az ember vizuális típus ezért könnyen azt gondolhatjuk, hogy az ilyen típusú feladatok könnyűek. Számunkra nem jelent gondot megtalálni egy autót a képen, vagy egy mozgó objektumot követni egy videón, esetleg egy rendszámtábla karaktereit leolvasni. Ha jobban belegondolunk, hogy a gép csak számokból álló mátrixot "lát", akkor könnyen megérthetjük, hogy ez nem is olyan triviális feladat, de a jobb megérthetőség érdekében a ***2.ábra*** segítséget nyújt.[2]



2. ábra. *A Számítógép számára csak egy számokból álló rácsként érzékeli a vizuális elemeket. [2]*

2. Módszerek

A rendszámtábla azonosító rendszerek általában két fő folyamatból tevődnek össze. Az első az, hogy detektálják a rendszámtáblát. A fejlesztők vonakodnak közzétenni a részleteket a kereskedelmi jellege miatt, viszont így is találni pár módszert a szakirodalomban a detektálással kapcsolatban.

A második folyamat a karakterfelismerés. Ez egy meglehetősen fejlett terület a számítógépes látás ágazatában, és számos technika áll rendelkezésre.

Számos módszer létezik mind detektálásra, mind karakterfelismerésre, de közülük csak néhányról olvashatunk a továbbiakban.

2.1. Rendszámtábla detektálása

A rendszámtábla azonosítás során, a tábla helyének meghatározása, illetve a normalizálása tekinthető a legnehezebb feladatnak. A nehézséget befolyásolja a táblatípusok méretbéli különbségei, valamint az alakjuk illetve színük sem egységesek. A rossz szögben lencsevégre kapott rendszámtáblák sem könnyítenek a feladat nehézségén. A felismerési folyamatot általában úgy tervezik, hogy csak bizonyos körülmények között működjön, mint például csak egy adott ország, adott típusbéli rendszámtábláit legyen képes csak felismerni, esetleg csak bizonyos perspektívában, illetve távolságban lévőket.

A szakirodalomban megfigyelhető, hogy rengeteg módszer alapját az éldetektáló algoritmusok adják, melynek használatával könnyebb lokalizálni a karaktereket tartalmazó területet.

A legtöbb módszer a tábla tulajdonságait használja ki, mint például a négyszögletű formáját. [3]

2.1.1. Függőleges élek keresése

M. Ahmed, M. Sarfaz, A. Zidouri és K G. AI-Khatib munkájában [4] a számtábla függőleges éleinek meghatározásához a Sobel operátort használják. Ezután a rendszámok típusának (szélességi és magassági arány) az ismeretét használják azon jelölt területek megkeresésére, ahol a rendszámtáblák nagyobb valószínűséggel vannak. [3]

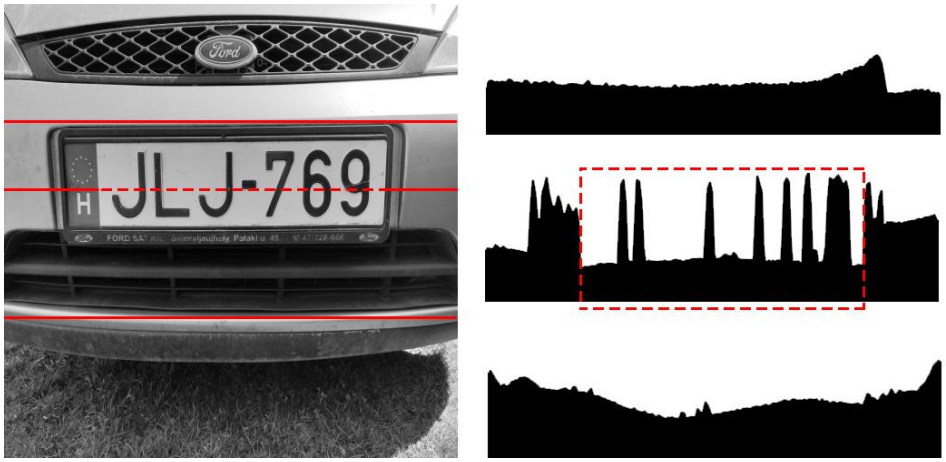
Ha szürkeárnyalatossá transzformált képen detektálni szeretnénk a megfelelő függőleges éleket, akkor az úgynevezett Sobel éldetektort érdemes használnunk, mivel ez a módszer rendelkezik a legjobb értékekkel. Az éldetektor által használt küszöbérték dinamikus, mert a rendszer automatikus értékeket vesz az algoritmusból. Megfigyelhető, hogy a legtöbb járművek több vízszintes vonallal rendelkeznek, mint a függőleges vonalakkal.

Ha sikeresen detektálásra kerül a két függőleges él a rendszámtáblán, akkor már a négy sarka is rendelkezésre áll a számtáblából, ami alapján már sikeresen lehet lokalizálni. Ez nagyban segít kinyerni magát a táblát a képről, még ha nem is teljesen tökéletes az alakzat.[4]

2.1.2. Lenyomat alapján való keresés

Túlnyomórészt a rendszámtáblák fekete karakterekkel, és fehér háttérrel rendelkeznek, így elmondható, hogy viszonylag sok fekete-fehér átmenet tömörül össze kis helyen. Ezt a tömörülést használja ki a lenyomat alapján történő detektálás. Lenyomatnak nevezzük a fényerő függvény minimális és maximális karakterisztikus sorrendjét, amelyet azokban a sorokban van jelent, ami keresztezi a rendszámtáblát.[5] A rendszámtáblánál lévő lenyomat sűrű vonások tartományát képezi, függetlenül a nézőpont változásától. Tehát módszer, a rendszámtábla detektálásához a bemeneti kép minden sorát ellenőrzi.

A következő ábrán jól megfigyelhető a tényerő függvény értékének ábrázolása, tehát minél sötétebb az adott pixel értéke annál nagyobb, illetve minél világosabb a pixel annál alacsonyabb az értéke. Továbbá az ábrán jól kivehető a lenyomat a középső vizsgált sornál, mivel az tartalmazza a rendszámtáblát, ellenben a másik két sor nem tartalmaz semmilyen kivetnivaló lenyomatot.



3. ábra. *Rendszámtábla lenyomat ábrázolása.*

2.1.3. Zárt téglalapok keresésének módszere

A módszer nagyban kihasználja a rendszámtábla alakjából származó tulajdonságokat, főként a téglalap alakú formáját. Elsőkörben detektálja a bemenő képen lévő zárt alakzatokat, majd ezeket sorba rendezi. A sorba rendezéssel az egészen apró zárt alakzatokat kizárhatjuk, hogy ne dolgozzunk a későbbiekben velük fölöslegesen, hiszen egészen biztos, hogy azok nem a rendszámtábla körvonalai.

A továbbiakban különböző szűréseket végez el a módszer zárt alakzatokon, míg megnem kapjuk a rendszámtábla helyét. A szűrések a korábban említett rendszámtábla tulajdonságokra specializálódnak. Ilyen szűrés lehet, hogy a zárt alakzat négy ponttal rendelkezzen, vagy négy oldal zárja közre az alakzatot.

2.2. Karakterfelismerés

Az optikai karakterfelismerés (Optical Character Recognition-OCR) szkennelt dokumentumok, vagy akár képen (a lényeg, hogy egy pixelekből álló képbe beágyazott szöveg legyen) lévő karakterek, szövegek kinyerésére lett kifejlesztve, ezáltal a számítógépes látás ágazatába lehet sorolni. Tehát röviden az OCR, karakterek felismerésére szolgál, legyen szó betűkről vagy számokról, esetleg speciális karakterekről, majd ezeket a kiolvasott karaktereket további műveletek elvégzésére használhatják. Ilyen művelet lehet például a szerkesztés, formázás, keresés, automatikusan lefordítás más nyelvekre vagy akár beszédre konvertálás. Az OCR technológiát széles körben használják a könyvek és dokumentumok digitális formába konvertálására, valamint lehetővé teszi a felhasználó számára a szöveg szerkesztését. [6]

V. K. GOVINDAN és A. P. SHIVAPRASAD a karakterek felismerése áttekintésében [7] kifejti a karaktersablon összehasonlításán alapú összehasonlítást, a karakterek sajátosságain alapuló, illetve a neurális hálózattal támogatott technikákat.

2.2.1. Karaktersablon összehasonlításán alapuló felismerés

Ez a módszer rendelkezik egy adatbázissal, ami a karakterek sablonját tartalmazza. Minden lehetséges bementi karakterhez külön sablon tartozik, amiket összehasonlít vele. Tehát az aktuális bementei karaktert minden egyes sablonnal összevet, és a legjobban egyezőt választja ki. Az előállított függvény ($s(I,T_n)$) visszatérési értéke mutatja meg, hogy a bemeneti karakter ($I(x,y)$) mennyire hasonlít a mintára ($T_n(x,y)$).

Néhány összehasonlító függvény:

- City block

$$s(I,T_n)=\sum_{i=0}^{w}\sum_{j=0}^{h}|I(i,j)-T_n(i,j)| \tag{1}$$

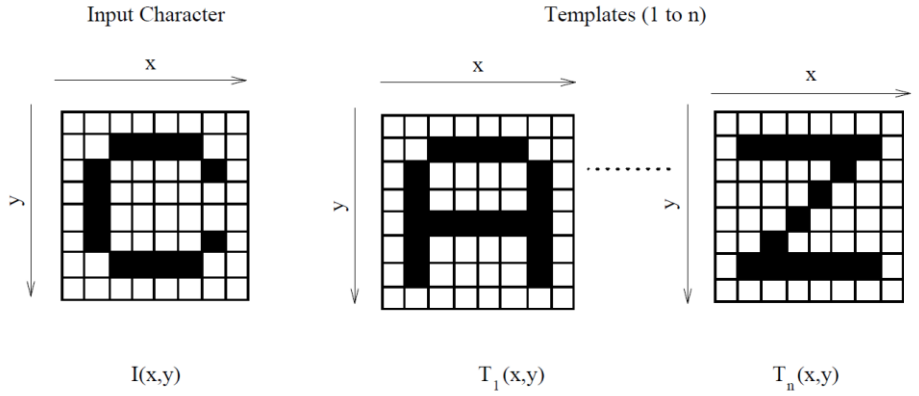
- Euclidean

$$s(I,T_n)=\sum_{i=0}^{w}\sum_{j=0}^{h}(I(i,j)-T_n(i,j))^2 \tag{2}$$

- Cross Corelation

$$s(I,T_n)=\sum_{i=0}^{w}\sum_{j=0}^{h}I(i,j)T_n(i,j) \tag{3}$$

A sablon alapú összehasonlítás igényel némi előfeldolgozást a bemenő képpel kapcsolatban. Ezek a módosítások általában vagy szürkeárnyalatossá való alakítás, vagy bináris képpé történő alakítás, esetleg valamilyen threshold művelet. [8]



4. ábra. *Karaktersablon összehasonlításán alapuló felismerés. [8]*

A módszer akkor igazán sikeres, ha a bemeneti karakterek és a tárolt sablonok azonosak, vagy legalább nagyon hasonlóak. Jelen feladatunkban, ez a módszer sikeresnek bizonyulhat, mivel a 326/2011. (XII. 28.) Korm. rendelet 11. melléklete tartalmazza a magyar rendszámokon lévő karakterkészletet, karaktersablont.

2.2.2. Karakterek sajátosságain alapuló felismerés

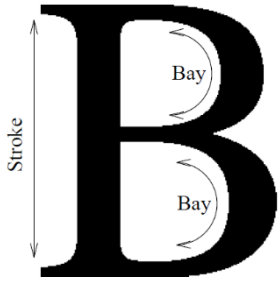
Ebben az esetben a legjobban hasonlító sajátossági jellemzők megtalálása alapján történik a felismerés. Első körben ezzel a módszerrel megkeressük a bementi karakterre jellemző sajátosságokat, amelyeket összehasonlítunk az adatbázisban eltárolt sajátosságokkal. A karakterek jellemzőinek kiválasztása többféle módon történhet, amelyek az alábbiak lehetnek:

- Teljes átalakítás, vagy sorozat bővítés
- Pontok statisztikai elhelyezkedésén alapuló sajátosságok
- Geometriai és topológiai jellemzők

Teljes átalakítás, vagy sorozat bővítés során a sajátosság vektor csökkentése mellett átalakító, arány és forgató algoritmust szükséges végrehajtani. Ilyen algoritmusok lehetnek a Fourier, Walsh, Haar, Hadamard, Hough transzformáció, a lánc-kód transzformáció vagy a főtengely algoritmus.

Pontok statisztikai elhelyezkedésén alapuló sajátosságok vizsgálata alapvetően kétféle módon történhet. Az egyik megoldás esetében zónákra osztjuk a mintát, de egy másik módszer lehet a karakteren belüli helyek, keresztező vonalat, illetve távolságiak vizsgálata. A vizsgálat nagy hátránya, hogy az eljárás során bemenő minták változnak, torzulnak.

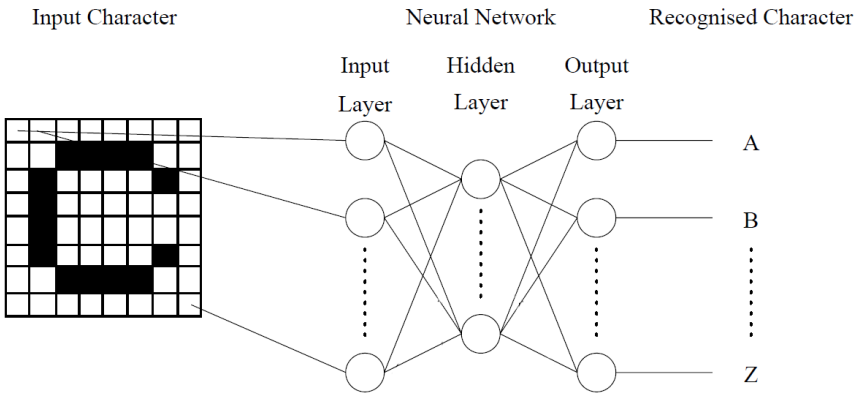
Geometriai vagy topológiai jellemzők alapján kiválasztott sajátosságok biztosítják a legnagyobb hatásfokot a betűtípus tulajdonságai, egyes képfeldolgozó algoritmusok által készített zajok, illetve a forgatással szemben.[8]



5. ábra. *A „B" karakter sajátosságai geometriai és topológiai szempont szerint. [8]*

2.2.3. Neurális hálózattal támogatott felismerés

A neurális hálózattal támogatott felismerés elsősorban osztályozási feladat elvégzésére alkalmas módszer. Az eljárás során a karakterek képeinek pixeleit használjuk a hálózat bemeneteként. Ezen kívül a karakterek tulajdonságaiból egyes sajátosságok is kiemelhetőek. A neurális hálózat architektúráját az alábbi ábra szemlélteti:



6. ábra. *Karakterfelismerés neurális háló segítségével. [8]*

A hálózatot különböző csomópontok építik fel, amelyeket neuronoknak nevezünk. A neuronokat függvények építik fel, kimenetük egy súlyozott összegből és egy az összegen végzett nem-lineáris függvényből állnak. A hálózat tanításakor szükséges paraméterezni a súlyokat. Maga a tanító algoritmusok célja a hibák minimálisra csökkentése, amely gradiens metódusok használatával érhető el.

A neurális hálózatnak három fő eleme van: a bemeneti réteg, a kimeneti réteg és egy rejtett réteg. A bemeneti réteget bemeneti vektorok alkotják, amelyek minden egyes neuronhoz hozzá vannak rendelve. A kimeneti rétegben a neuronnak felismerendő karakterhez tartoznak.

Amikor a bemeneti vektornak értéket adunk, akkor az a kimenet lesz a legerősebb, amelyhez az azonosított karakter tartozik.[8]

3. Használt technológiák

A nyílt forráskódú technológiák megjelenésével a számítástechnika világ egy új szintre emelkedett.

A zárt forráskódú programokkal ellentétben nem korlátozzuk be a tovább fejleszthetőségi lehetőségeket, mivel az open source világában minden ingyenesen elérhető és a programkódok szabadon

hozzáférhetőek mindenki számára. A világ minden tájáról lelkes programozók önkéntesen fejlesztik tovább a meglévő technológiákat, a saját igényeikre alakítva, vagy ha nagyközönség érdekeit is szolgálja a fejlesztés, akkor szigorú ellenőrzés után beleintegrálják a meglévőbe, ellentétben a zárt forráskódú programokkal, ahol csak a cég emberei dolgoznak a szoftveren.

A nyílt forráskódnak köszönhetően alakultak ki a következő technológiák, melyek jelentősen megkönnyítik a rendszámtábla azonosító rendszer létrehozását.

3.1. Python

A Python egy magas szintű, általános célú programozási nyelv, mely rendkívül dinamikus. A széleskörű elterjedését többek között, köszönheti annak, hogy bővíthető, ingyenes, valamint egyszerű szintaxissal rendelkezik. Előnyei között élen szerepel a platformfüggetlensége, tehát használhatjuk különböző Unix változatokon, MaxOS-en és valamennyi Windows változatokon is. A nyelv a korábban említett nyílt forráskódú technológiák táborát bővíti, ennek köszönhetően folyamatosan fejlődik a lelkes fejlesztők, és felhasználók által. [9]

A Python további előnyeiről, és rejtelmeiről is olvashatunk, Gérard Swinnen magyarul is megjelent *Tanuljunk meg programozni Python nyelven* című könyvében.

3.2. OpenCV

Az Open source Computer Vision röviden csak OpenCV, egy olyan nyílt forráskódú függvénykönyvtár, ami a számítógépes látásra lett kifejlesztve. A könyvtár C és C++ programnyelven íródott, és többek között Windows, Linux és Mac OS X operációsrendszerekkel kompatibilis. Elsődleges interfésze a C++, de fejlesztik több különbféle nyelvekre, köztük Python-ra, Ruby-ra, valamint Matlab-ra is.

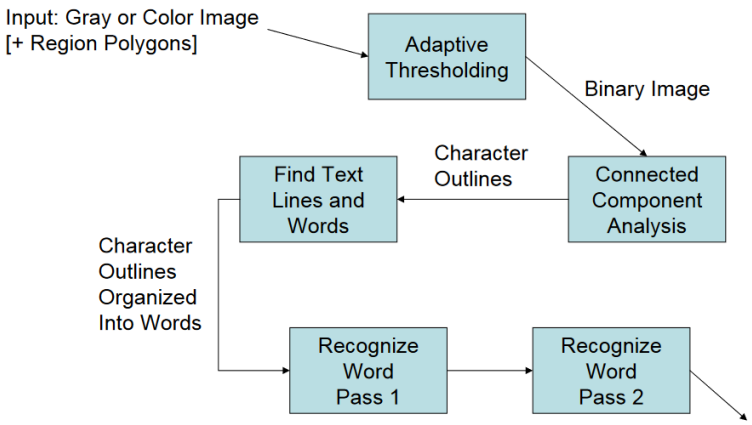
Elsődleges céljai között szerepel, hogy a használói számára biztosítson egy egyszerűen használható infrastruktúrát a számítógépes látás témakörében, amely segít az alkalmazások gyors fejlesztésében.
Az OpenCV könyvtár több mint 500 függvénnyel rendelkezik, melyek számos területet lefednek a számítógépes látás területén, ideértve az orvosi képalkotást, a robotikát és többek között a termékvizsgálatot a különféle gyárakban.[2]

Tehát az OpenCV függvénykönyvtár jelentősen megkönnyíti a rendszámtábla azonosítás menetét, így ezért is élek az általa nyújtott lehetőségekkel az áttekintő cikkben.

3.3. Tesseract-OCR

A Tesseract egy nyílt forráskódú, OCR motor, melyet C és C++ nyelven fejlesztettek a Hewlett-Packart (HP) laboratóriumban 1985 és 1996 között. 2007-től a Google folytatja a továbbfejlesztését és karbantartását. Szürkeárnyalatos vagy színes képet vesz bemenetként és szöveges formátumba adja vissza. Kezdetben csak .tiff formátumú képeket támogatta, de mára már png, vagy jpg illetve sok más típusú képet is támogat. A tesseract több nyelven is képes felismerni a szöveget, mint például angol, svéd, dán stb. Windows, Ubuntu és más operációs rendszereken is működik, illetve már mobil platformokon is használják, mint például Android és IOS.[10]

A következő ábra szemlélteti a Tesseract architektúráját.



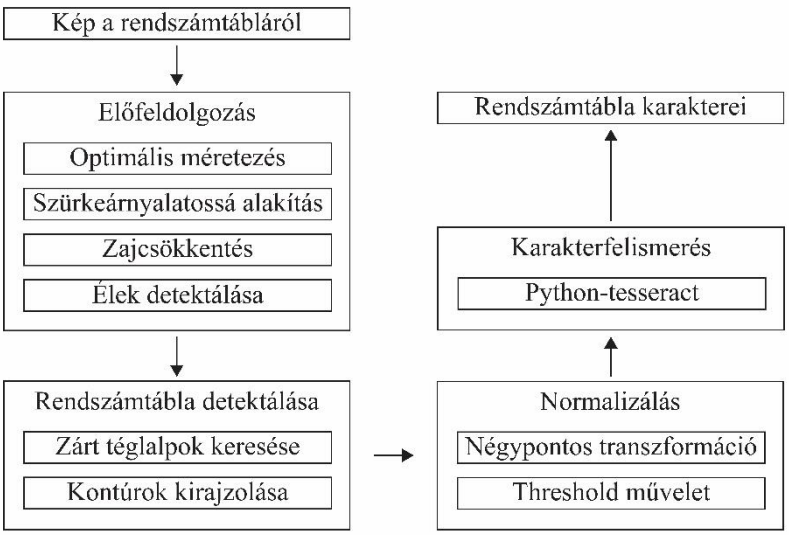
7. ábra. *Tesseract-OCR architektúrája. [11]*

3.3.1. Python-Tesseract

Python-Tesseract egy optikai karakterfelismerő (OCR) eszköz a pythonhoz, vagyis képes felismerni és kiexportálni karakterek formájában a képeken lévő szöveget. A Python-Tesseract egy úgynevezett csomagolás (wrapper) a korábban említett Google által fejlesztett Tesseract-OCR-hez. Több bementei formátummal is képes dolgozni, mint például JPEG, PNG, GIF, BMP és TIFF. [12]

4. Rendszer felépítése

A Rendszámtáblafelismerő rendszer struktúráját a következő ábra szemlélteti, melyben jól átlátható a különböző fázisok folyamatai.



8. ábra. *A rendszámtábla azonosító rendszer felépítése.*

4.1. Előfeldolgozás

Az előfeldolgozásban képmanipulációs módosításokat végzünk el a képen, és végül éldetektált formában adjuk át a rendszámtábla detektáláshoz.

4.1.1. Optimális méretezés

Az előfeldolgozás első lépése a bejövő képek optimális méretre való szabása. Az átméretezés segít elkerülni a nagy felbontású képekkel való problémákat, viszont ilyenkor ügyeljünk arra, hogy a rendszámtábla az átméretezés után is a képen maradjon.

9. ábra. *Az optimális méretre való méretezés után.*

Alapértelmezetten 500x500-as méretre célszerű méretezi a bejövő képeket. A méretezést a cv2.resize nevű függvénnyel lehet megvalósítani ahol a cv2 az OpenCV függvénykönyvtárra utal.

4.1.2. Szürkeárnyalatossá alakítás

A következő lépésben az átméretezett képet szürkeárnyalatossá kell alakítani. Ez a lépés minden képfeldolgozási eljárásban gyakori, mivel így már nem kell foglalkozni többé a színes részekkel, ezáltal felgyorsulnak a folyamatok.

10. ábra. *Szürkeárnyalatossá alakítás után.*

Szintén az OpenCV függvénykönyvtár egyik függvényével egyszerűen kivitelezhető az átalakítás. A cv2.cvtColor () metódust használják, ha egy képet az egyik színmódról konvertálni kell egy másikra

színmódra. Több mint 150 színmód-átalakítási módszer érhető el az OpenCV-ben, de jelen esetben a szürkeárnyalatos kell melyet a cv2.COLOR_BGR2GRAY-el lehet elérni.

4.1.3. Zajcsökkentés

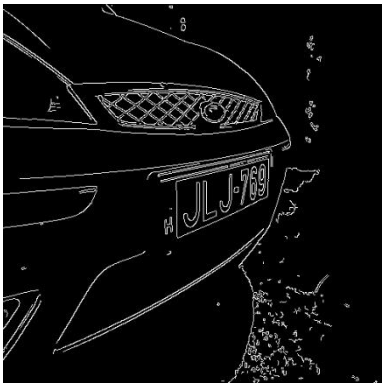
A szürkeárnyalatossá alakítás után sor kerül az úgynevezett zajcsökkentésére, mely fontos szerepet tölt be a haszontalan részek kiszűrésében. Ez a bilateral szűrő segítségével valósul meg, mely egy enyhe elmosódó hatást rak a képre.

11. ábra. *Zajcsökkentés után.*

A bilateral szűrő, paraméterezése a következő: cv2.bilateralFilter (forráskép, pixel átmérője, sigmaSzín, sigmaTér). Az utolsó két paraméterrel lehet kísérletezni, hogy mely értékük az optimálisabb, viszont ügyelnünk kell rá, hogy a rendszámtábla ne mosódjon el.

4.1.4. Élek detektálása

Az előfeldolgozás utolsó lépésben az élek detektálását végezzük el. Ennek számos módja van, viszont a legegyszerűbb és egyben a legnépszerűbb módszer az OpenCV módszerének használata.



12. ábra. *Élek detektálása.*

Az élek detektálásához a Canny éldetektort használjuk, ami az OpenCV függvénykönyvtárában található. Első paramétereként megadjuk a képet, majd a második paraméterke az első küszöbérték a hiszterézis eljárás számára, a harmadik paraméter pedig a második küszöbérték.

4.2. Rendszámtábla detektálása

A rendszámtábla azonosítására a már említett téglalapok keresésének módszerét alkalmazzuk, majd az optimalizált képre kirajzoljuk a kontúrokat. A detektálást az előfeldolgozó egység utolsó lépésének az eredményén alkalmazzuk, tehát azon a képen, ahol csak az élek vannak jelen.

4.2.1. Zárt téglalapok keresése

A körvonalakat a cv2.findContours nevű függvénnyel találhatjuk meg. Több zárt alakzatot fog találni, és az első tízet csökkenő sorrendbe tesszük. Nagy valószínűséggel a rendszámtáblánk is közte lesz a tíz alakzat között. További szűrések kapcsán leellenőrizzük, hogy mely alakzatoknak van négyszög alakú körvonala, négy oldallal körbe zárva.

Ha például megtalálja egy egyenes vonal kontúrját, akkor nincs szüksége az összes pontra a vonalon, elég csak a kezdőpont illetve a végpont. Erre szolgál a cv2.CHAIN_APPROX_SIMPLE ami eltávolítja az összes redundáns pontot a vonalon ezzel sok memóriát takarítva meg.

4.2.2. Kontúrok kirajzolása

A kontúrok rajzolásához a cv2.drawContours függvényt kell használni. Használható bármilyen alak rajzolására is, ha megvannak a határpontjai, de számunkra négyzeteket fog rajzolni, feltéve, ha megtalálja a rendszámtáblát. Ez a lépés nem feltétlenül szükséges a helyes működéshez, inkább csak az ellenőrzés miatt lényeges.

13. ábra. *Kontúrok kirajzolása.*

4.3. Normalizálás

4.3.1. Négypontos transzformáció

A négypontos transzformáció célja egy egységes frontális nézet előállítása, még a más perspektívából készült képeknél is.

14. ábra. *Négypontos transzformáció eredménye.*

A transzformáció menete a következő:

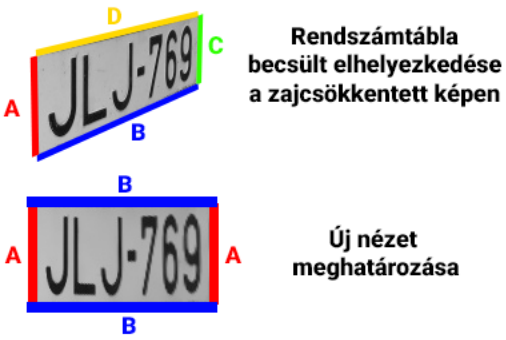
1. A becsült téglalapnak, melyet a rendszámtábla azonosításánál találtunk meg, a 4 sarok pontját sorba rendezzük a pontokkal való munkavégzés egyszerűsítése miatt. A sorba rendezés igy fog kinézni: (T op left, Top right, Bottom right, Bottom left)
2. A leghosszabb szélesség meghatározása:

$$AB = d = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2} \tag{4}$$

3. A legnagyobb magasság meghatározása:

$$AB = d = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2} \tag{5}$$

4. Új keret készítése
5. A rendszámtábla beillesztése az új keretbe OpenCV segítségével



15. ábra. *A transzformáció szemléltetése az oldalak segítségével.*

4.3.2. Threshold művelet

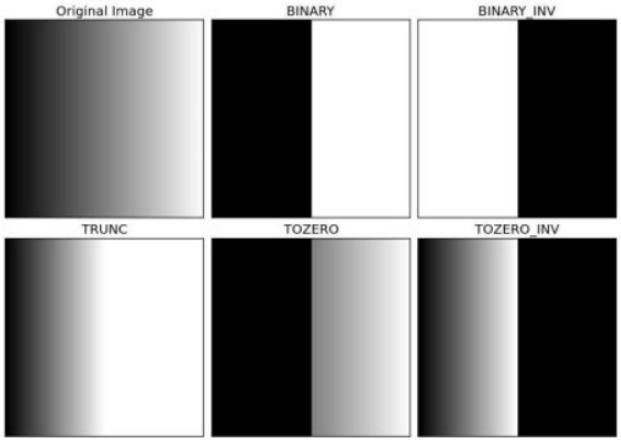
A normalizálás témakörébe tartoznak az úgynevezett threshold műveletek, melyek kontrasztossá teszik a rendszámtáblát ezáltal megkönnyítik a karakterfelismerést.

A cv2.threshold() nevezetű függvénnyel tudunk küszöbérték műveletet végrehajtani. Az első argumentumának egy szürkeárnyalatos képnek kell lennie, majd a második paraméter a küszöbérték, amelyet

a pixelértékek osztályozására használnak. A harmadik argumentum a maxVal, amely képviseli azt az értéket, amit meg kell adni, ha a pixelérték nagyobb, mint a küszöbérték. Az OpenCV különböző küszöbstílusokat biztosít, és ezt a függvény negyedik paramétere határozza meg.

Típusok:

- cv2.THRESH_BINARY
- cv2.THRESH_BINARY_INV
- cv2.THRESH_TRUNC
- cv2.THRESH_TOZERO
- cv2.THRESH_TOZERO_INV



16. ábra. *Threshold műveletek ábrázolása.* [13]

4.4. Karakterfelismerés

4.4.1. Python-tesseract

Az első paraméterében kapja meg a normalizált képet a könnyebb keresés érdekében. Az „-l eng” paraméterrel adjuk meg, hogy angol karaktereket keressen a képen. Mivel a magyar rendszámokon nincs ékezetes betű így számunkra optimális. Az ezt következő „--oem N” paraméternek 4 opciója van mellyel az OCR motor módját tudjuk meghatározni:

- 0 = Eredeti Tesseract.
- 1 = LSTM.
- 2 = Tesseract + LSTM.
- 3 = Alapértelmezett, a rendelkezésre álló adatok alapján.

Az utolsó paraméter a „--psm N” ahol az N változó értéke egy 0 és 13 közötti szám lehet. Íme közülük néhány:

- 6 = Kezelje úgy a képet, hogy egy szövegblokk található rajta.
- 7 = Kezelje úgy a képet, hogy egyetlen sor szöveg található rajta.
- 8 = Kezelje úgy a képet, hogy csak egyetlen szó található rajta.
- 10 = Kezelje úgy a képe, hogy csak egyetlen karakter található rajta.
- 13 = Kezelje a képet egyetlen szöveges sorként, [14]

5. Eredmények

Az alábbi táblázatban láthatóak az eredmények a karakterek felismerésével kapcsolatban. Az esetek nagyrészében a program 100%-os eredményt produkált, míg máshol több karaktert talált a kelleténél, vagy helytelenül ismerte fel.

1. táblázat. *Eredmények*

Kontúrozott kép	Normalizált kép	Kiolvasott karakte-rek	Eredmény
		JLJ-769	100%
		JLJ-769	100%
		JLJU-769	85%
		JILJ-769	85%
		JLJ-769	100%
		JLu-749	71%

6. Összefoglalás

A publikációban bemutatásra kerültek azok a módszerek és technológiák, amelyek segítségével létre-hozható egy olyan rendszámtábla azonosító rendszer, ami képes detektálni, illetve azonosítani is egy

digitális képen szereplő autó rendszámtábláját. Az eredményeken jól látható, hogy nem jelent gondot a rossz perspektívából elkapott számtáblák azonosítása sem.

7. Köszönetnyilvánítás

A cikkben ismertetett kutató munka az EFOP-3.6.1-16-2016-00011 jelű „Fiatalodó és Megújuló Egyetem – Innovatív Tudásváros – a Miskolci Egyetem intelligens szakosodást szolgáló intézményi fejlesztése" projekt részeként – a Széchenyi 2020 keretében – az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Irodalom

[1] Nixon, M., Aguado, A.: *Feature extraction and image processing for computer vision*, Academic press, 2019. **https://doi.org/10.1016/B978-0-12-814976-8.00003-8**

[2] Bradski, G., Kaehler, A.: *Learning OpenCV: Computer vision with the OpenCV library*, O'Reilly Media, Inc., 2008.

[3] Basheer, H. N. A.: *Sudanese license plate recognition System*, 2017, PhD Thesis, University of Khartoum.

[4] Ahmed, M. J., et al.: *License plate recognition system*, In: 10th IEEE International Conference on Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003. IEEE, 2003. p. 898-901.

[5] Kwaśnicka, H., Wawrzyniak, B.: *License plate localization and recognition in camera pictures*, In: 3rd Symposium on Methods of Artificial Intelligence, 2002. p. 243-246.

[6] Herusutopo, A., et al.: *Recognition design of license plate and car type using Tesseract OCR and EmguCV*. CommIT (Communication and Information Technology) Journal, 2012, 6.2: 76-84. **https://doi.org/10.21512/commit.v6i2.573**

[7] Govindan, V. K., Shivaprasad, A. P.: *Character recognition—a review*. Pattern recognition, 1990, 23.7: 671-683. **https://doi.org/10.1016/0031-3203(90)90091-X**

[8] Setchell, C. J.: *Applications of computer vision to road-traffic monitoring*. 1998. PhD Thesis. University of Bristol.

[9] Swinnen, G.: *Tanuljunk meg programozni Python nyelven*. 2005.

[10] Liaqat, A. G.: *Mobile real-time license plate recognition*. 2011.

[11] Smith, R.: *Tesseract ocr engine*. Lecture. Google Code. Google Inc., 2007.

[12] Kurlekar, S.: *Reading device for blind people using Python, OCR and GTTS*.

[13] https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

[14] https://tesseract-ocr.github.io/tessdoc/ImproveQuality