SZÉCHENYI ISTVÁN EGYETEM
GÉPÉSZMÉRNÖKI, INFORMATIKAI ÉS VILLAMOSMÉRNÖKI KAR

Informatika Tanszék

Rendszám tábla felismerés

Gépi Látás (GKNB_INTM038)

Szép Zsófia Szonja, KOEZRR

Gazdaságinformatikus

Github Repository: https://github.com/szepzsofia/GepiLatas_RendszamFelismero

2022.11.27.

Tartalom

Bevezetés

A mai rohamosan fejlődő világban a járművek száma is gyors tempóban növekszik. Azonosításuk lassan már elengedhetetlen, azonban nagy számuk miatt ez egyre nehézkesebb. Minden gépjármű rendelkezik egy egyedi azonosítóval. Ez az azonosító általában egy alfanumerikus karaktersorozat, másnéven forgalmi rendszámtábla. A rendszámtábla leolvasása a legegyszerűbb módja a járművek azonosítására. Emiatt napjainkban egyre nagyobb az igény az ilyen típusú felismerő rendszerekre. Használják a rendszámtábla felismerő rendszereket a rendőrök a traffipaxok során, a parkolóházakban és az emberek az okosotthonokban.

A továbbiakban a rendszámtábla detektálásáról és fényképről leolvasásáról fog szólni.

Elméleti áttekintés

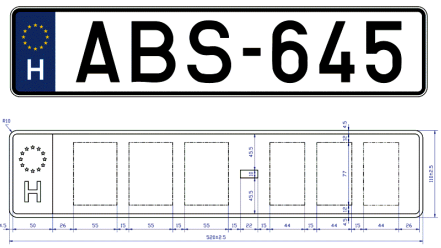
1. Rendszámtábla ismertetése

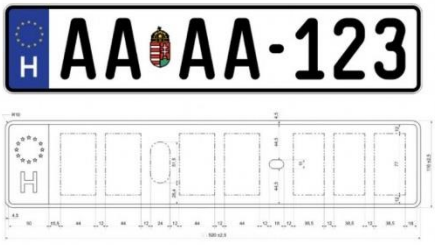
A forgalmi rendszám a közúton használt járművek egyedi azonosítója a forgalomba. Ezt a jármű hátuljára mindenhol kötelező felszerelni, sok helyen az elejére is. Magyarországon például első és hátsó rendszámtábla felszerelése is kötelező.

Nálunk jelenleg a gépjárműveink azonosítására egy 6 vagy 7 karakteres sorozatot használunk, ami általános esetben 3 vagy 4 betű utána 3 szám felosztásban követik egymást fehér alapon fekete karakterekkel. A rendszámtáblán megtalálható még az Unió 12 csillagos emblémája és egy fehér H betű kék színű sávban. 2022. július 1-től pedig már a magyar címer is megtalálható rajtuk.

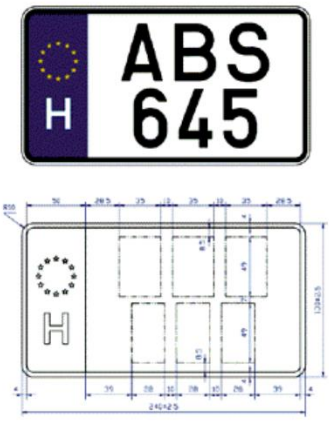
A 326/2011. (XII. 28.) Korm. rendelet 11. melléklete alapján, Magyarországon öt különböző rendszámtábla típust különböztetünk meg, melyek az „A”, „B”, „C”, „D”, illetve az „E” jelzést kapták. A karakterek elrendezését tekintve csoportosíthatunk egysoros, kétsoros továbbá háromsoros rendszámtáblákat is, valamint ezek a csoportok méretükben is különböznek. Emellett fontos megemlíteni, hogy bérfuvarozók esetén sárga, elektromos járművek esetén zöld az alapszín.

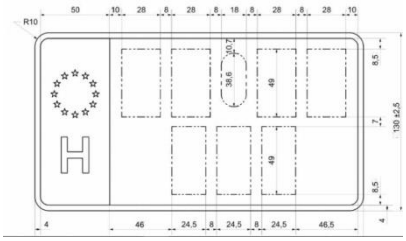
- „A” és „D” típusú rendszámtábla rajzai



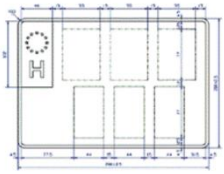


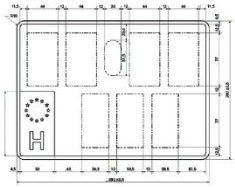
- „B” típusú rendszámtábla rajzai



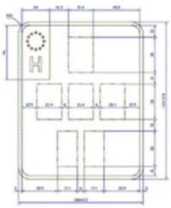


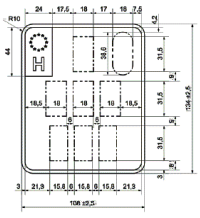
- „C” típusú rendszámtábla rajzai





- „E” típusú rendszámtábla rajzai





Hazánkban is van lehetőség egyedi rendszámtábla készítésére, melyekre a 326/2011. (XII. 28.) Korm. rendelet 58. § (1) bekezdésében leírtak vonatkoznak. Az

egyedi rendszámoknak legalább 3 folyamatos betűjelet és legalább 1 folyamatos számjegyből, együttesen 7 jelből kell állnia.

2. *Módszerek*

A rendszámtábla felismerő rendszerek általában két fő folyamatból tevődnek össze. Az első az, hogy detektálják a rendszámtáblát, a második folyamat pedig a karakterfelismerés.

2.1. *Rendszámtábla detektálás – Canny detektor*

Manapság az egyik legelterjedtebb és megbízhatóbb éldetektálási módszer. Minden valódi élt detektál és megbízhatóan szűri a „hamis” éleket. Minden élt pontosan egyszer jelez és ezeket pontosan lokalizálja.

Több lépéses kereséssel garantálja a megbízhatóságot:

- Szürke árnyalati konverzió:

  Első lépés az élkereséshez, amelynek során a bemeneti színes képet fekete-fehér képpé alakít át. Szürkeárnyalatos fotók esetén leggyakrabban a [0, 255] intenzitástartomány használt. A 0 érték jelenti a feketét, a 255 a fehér színt. A köztes értékek a feketéből fehérbe tartó szürke átmenetet jelentik. Az eredményt felhasználva tovább halad a kereső.

- Gauss simítás:

  A képpontok közelebb kerülnek környezetük átlagához, azaz a kép „simább” lesz. Ezt Gauss függvénnyel érjük el. Átlagoljuk a pixelek egy kis környezetét és feltételezzük, hogy a kép lokálisan homogén, a zaj pixelenként független. Fő tulajdonságai: σ (szigma): a Gauss függvény szórása megadja, hogy a középponttól távolodva milyen gyorsan csökkennek a súlyok. A Gauss függvény a teljes síkon értelmezett.

- Differenciaszámítás:

  Két pontbeli gradienseinek kiszámítása (közelítése). Általában minimális méretű környezetre törekszünk a differenciálás során. A függvény szélén lévő értékek kitöltetlenek maradnak.

- Nem-maximumvágás:

A nem-maximumvágás a Canny módszer által alkalmazott élvékonyítási technika. Minden élpontról megvizsgálja, hogy a gradiensének irányában lévő szomszédok közül az-e a legnagyobb hosszértékű. Ha igen, meghagyja élpontnak, ha pedig nem, akkor a továbbiakban nem tekinthető élpontnak.

- Kettős küszöbölés:

  A küszöbölés első lépéseként kiválasztunk egy adott küszöböt és ezután a kép egyes pixeleihez tartozó gradiens értékeket ezzel hasonlítja össze. Az értékek tartományától függően a továbbiakban a nem fontos információkat el is felejthetjük, például az elég kis értékeket (a küszöbnél alacsonyabbakat) mind 0-ra cserélhetjük, vagy a nagyokat mind 255-re. A kép megjelenítésekor továbbra is a [0; 255] intervallumból vannak értékeink.

- Hiszterézis küszöbölés:

  Feltéve, hogy már minden élpontunkat besoroltuk gyenge vagy erős élpontnak, a gyenge élpontoknak megvizsgáljuk a környezetét. Különböző képeken azonos értékek hatékonysága igencsak különböző lehet, de feltesszük, hogy találunk olyan küszöböket, amik jók. Ha a szomszédok között erősél pontot találunk, akkor belőle is erőset csinálunk. Ezt addig csináljuk amíg minden gyenge élpontra teljesül, hogy vagy erős élpontot csináltunk belőle, vagy beláttuk, hogy egy olyan gyenge él egy pontja, amely nem csatlakozik sehol sem erős élhez.

2.2. Karakterfelismerés

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 Google által fejlesztett Tesseract-OCR-hez. Több bemeneti formátummal is képes dolgozni, mint például JPEG, PNG, GIF, BMP és TIFF.

Fejlesztői dokumentáció

A feladat megvalósításához az órán is tanult Python programnyelvet és a hozzá kapcsolódó OpenCV kiegészítő könyvtárat használtam.

A kód írását a program sikerességéhez tartozó Python modulok importálásával kezdtem.

```
import cv2
import imutils
import pytesseract
import messagebox

pytesseract.pytesseract.tesseract_cmd = 'C:\Program Files\Tesseract-OCR\\tesseract'
```

A program megírását a már előre lementett autók képeinek beolvasásával folytattam. Majd a kép feldolgozásának első lépéseként az imutils.resize() segítségével optimális méretre szabtam a képet. Ennek köszönhetően el tudtam kerülni a nagy felbontású képek okozta problémát.

Ezután következett az átméretezett kép szürkeárnyalatossá tétele. A színes színmódról a szürkeárnyalatosra való konvertáláshoz a cv2.cvvtColor() metódust használtam, azon belül pedig a cv2.COLOR_BGR2GRAY- el tudtam ezt elérni.

A zajcsökkentésre azért van szükség, hogy a számunkra haszontalan részecskéket ki tudjuk szűrni. Ezt a bilateral szűrő közreműködésével tudjuk megvalósítani, a szűrő használata után a képen egy kisebb elmosodást láthatunk. A kód a következő:cv2.bilateralFilter(forráskép, pixel átmérő, szigmaszín, szigmatér).

Az élek detektálásához a Canny módszert alkalmaztam. Itt szintén az első paraméter a kép, a második és a harmadik pedig a hiszterézishez szükséges alsó felső küszöbérték.

A következő lépéseknél már azt a képet használom, amelyen csak az élek szerepelnek. A kontúrokat a cv2.findContours(forráskép, kontúrlekérési mód, kontúr közelítési módszer) függvény segítségével kerestem meg. A függvény több zárt formát is talált és az első tízet csökkenő sorrendben rangsorolta. Nagyon valószínű, hogy a mi rendszámunk is e tíz alakzat között lesz. További szűréssel kombinálva ellenőrizzük, hogy mely alakzatok négyszögletes körvonalúak, négy oldallal körbezárva. Például egy egyenes körvonalának megkereséséhez nincs szükség az egyenes minden pontjára, csak a kezdőpontra és a végpontra. Erre való a

cv2.CHAIN_APPROX_SIMPLE (harmadik paraméter), amely eltávolítja az összes redundáns pontot a vonalról, így sok memóriát takarít meg.

Kontúrok rajzolásához használtam a cv2.drawContours(forráskép, kontúrok, kontúrok indexe) függvényt. Bármilyen alakzatot lehet vele rajzolni, amíg vannak határpontjai, de ebben az esetben addig rajzol négyzeteket, amíg megtalálja a rendszámot. Ez a lépés nem feltétlenül szükséges a megfelelő működéshez, csak az ellenőrzéshez.

```
kep = cv2.imread('test.jpg')
kep = imutils.resize(kep, width=300) #Kép átméretezése
cv2.imshow("Alap kep", kep) #Megjelenik az alap kép átméretezve

szurke_kep = cv2.cvtColor(kep, cv2.COLOR_BGR2GRAY) #Kép színének szürkeárnyalatossá tétele
cv2.imshow("Szurke arnyalatos kep", szurke_kep) #Szürkeárnyalatos kép megjelenítése

szurke_kep = cv2.bilateralFilter(szurke_kep, 11, 17, 17) #Kép simítása
cv2.imshow("Simitott kep", szurke_kep) #Simított kép megjelenítése

szel = cv2.Canny(szurke_kep, 30, 200) #Kép éleinek detektálása
cv2.imshow("El detektalas", szel)

kontur,new = cv2.findContours(szel.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE) #Összes kontur keresése
kepmas=kep.copy()
cv2.drawContours(kepmas,kontur,-1,(0,255,0),3) #Konturok kirajzolása
cv2.imshow("Konturok",kepmas)

kontur = sorted(kontur, key = cv2.contourArea, reverse = True) [:10] #Konturok kikeresése, legjobb 10 kiválasztása
screenCnt = None
kepmas2 = kep.copy()
cv2.drawContours(kepmas2,kontur,-1,(0,255,0),3)
cv2.imshow("Top 10 kontur",kepmas2)
```

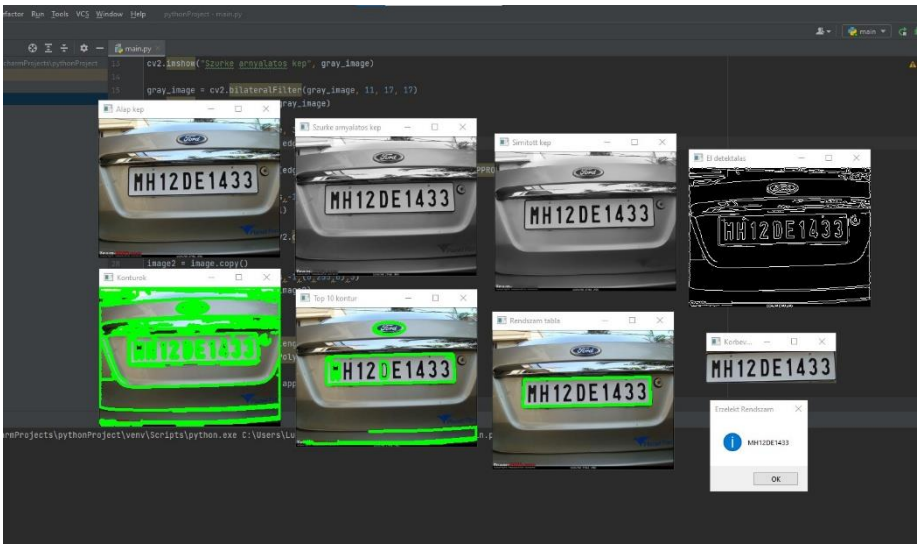
A következő lépésként a kiemelt kontúrok közül megkerestem a téglalapot, körbevágtam es kimentettem egy újonnan létrehozott png típusú képfájlként. Létrehoztam a sarokpontok segítségével egy új képet, amit aztán lementettem a .png fájlba.

```
kivagott = './7.png'
cv2.imshow("Korbevagott", cv2.imread(kivagott))

i=7
for c in kontur:
        perimeter = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.018 * perimeter, True)
        if len(approx) == 4:
                screenCnt = approx

        x, y, w, h = cv2.boundingRect(c)
        new_img = kep[y:y + h, x:x + w]
        cv2.imwrite('./' + str(i) + '.png', new_img)
        i += 1
        break

cv2.drawContours(kep, [screenCnt], -1, (0, 255, 0), 3)
cv2.imshow("Rendszam tabla", kep)


rendszam = pytesseract.image_to_string(kivagott, lang='eng')

messagebox.showinfo(title="Erzekelt Rendszam", message=rendszam)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Tesztelés

3 tesztelést végeztem 3 különböző képen.

Az első esetben sikeresen lefutott a program, ez az alábbi képen látható.



A másik két kép esetén nem ment végbe a program. Először a kötőjel okozott problémát. A Python-Tesseract nem ismerte fel ezt a karaktert. A második esetben pedig túl sok volt a detektált él és a nem megfelelő téglalapot vágta ki a program. Ezen problémák a későbbiekben megoldásra kerülnek.