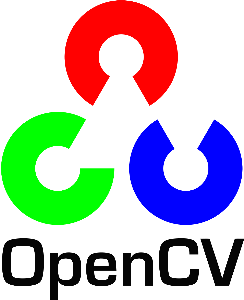
**Dokumentáció**

Az alábbi dokumentáció egy olyan programról lesz, ami egy egyszerű algoritmus alapján, képes felismerni korlátozott számú, előre meghatározott közlekedési táblát. Az algoritmus alapja a négyzetek felismerése, az érzékelt útjelző tábla szegmensekre osztása, a szegmens értékének meghatározása és ez alapján döntéshozatal.

**Szükséges hardware és software a program működéshez:**

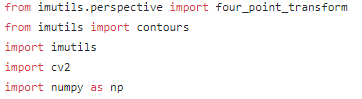
* Kamera
* OpenCV library
* ****Numpy
* Python
* Imutils

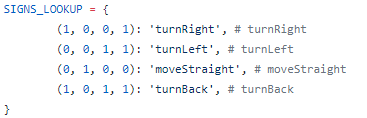
**Felhasznált külső könyvtárak:**

* cv2:OpenCV elérés
* Numpy : kiegészítő csomag a Python programozási nyelvhez magas szintű matematikai függvénykönyvtárral.
* Imutilis : kényelmi funkciók sorozata, ami egyszerűbbé teszi a képfeldolgozási műveleteket, mint péládul a fordítás, a forgatás és az átméretezés.

**Az algoritmus:**

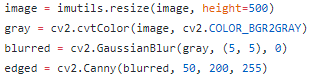
**1.)** elvégezzük a modul hívást.



**2.)** definiáljuk a táblák szegmenseinek értékeit,hogy később fel tudja őket ismerni a program.

**3.)** Elindíjtuk a kamerát:

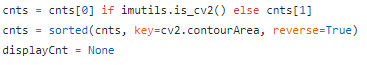
C:\Users\Noel\Desktop\Nocinak\3.png

**C:\Users\Noel\Desktop\Nocinak\4.png4.)** Előkészítjük a kamera képét, azaz elvégezzük a szürkeárnyalat konverziót, méretre szabjuk, elhomályosítjuk és elkészítjük az ’edge map-ot’(szélek feltérképezése).

**5.)** Megkeresem a kontúrokat és csökkenő sorrendbe rendezem.

C:\Users\Noel\Desktop\Nocinak\6.png

**6.)** Az is\_cv2() és is\_cv3() egyszerű függvények ,amiket arra használunk, hogy automatikusan meghatározzuk az OpenCV verzióját a jelenlegi környezetnek. A cnts[0] és a cnts[1] kontúrokat tartalmaz.



**7.)** Iterációk segítségével átmegyünk a kontúrokon, és megközelítjük őket.

**C:\Users\Noel\Desktop\Nocinak\8.png**

**C:\Users\Noel\Desktop\Nocinak\9.png**

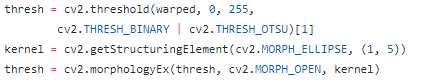
peri – a képen lévő tárgyak tulajdonságai és helyzetei.

**C:\Users\Noel\Desktop\Nocinak\10.png8.)** Ha a kontúrnak négy sarka van, akkor megtalátuk a négyzetet.

**9.)** A tábla határvonalait használva perspektíva átalakítást végzünk. Egy gyakori perspektívát az úgynevezett "birds eye view"-ot alkalmazzuk ,ami fentről lefele való nézetet biztosít.

**C:\Users\Noel\Desktop\Nocinak\11.png**

**C:\Users\Noel\Desktop\Nocinak\12.png10.)** Egy piros négyzettel jelöljük a táblát a videó képén.

**11.)** A szürke árnyalataival megjelenített képet(warped) bináris képpé konvertáljuk(thresholding). Majd morfológiai utasítások segítségével letisztítom.

cv2.THRESH\_OTSU – automatikusan kiszámolja a küszöbértéket a kép hisztogrammjából.

**12.)** subHeight, subWidth érték megadása és integerré alakítása.

(roiH, roiW) = roi.shape

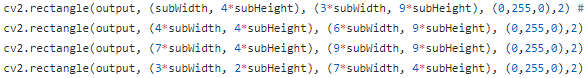
subHeight = thresh.shape[0]/10

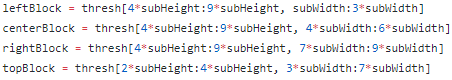
subWidth = thresh.shape[1]/10

C:\Users\Noel\Desktop\Nocinak\14.png

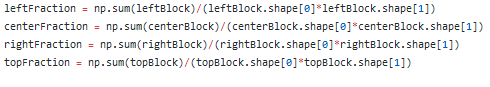
np.divide – skaláris az eredménye, a bemeneti adatoktól elhatárolódva adja meg a kimeneti adat típusát.

**13.)** Megjegyezzük a tábla határait a képről.



**14.)** Kivágunk négy blokkot a tábláról,amiket majd vizsgálni fogunk.

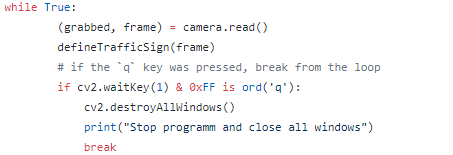
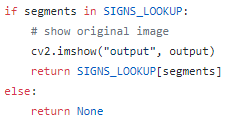
**15.)** Kiszámoljuk a négy részegység által meghatározott értékeket (aktív pixelek összege/összes pixel száma).



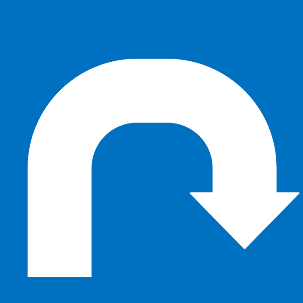
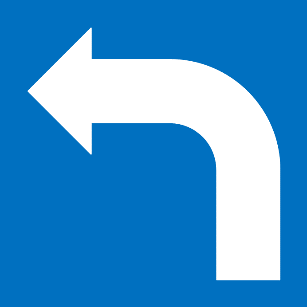
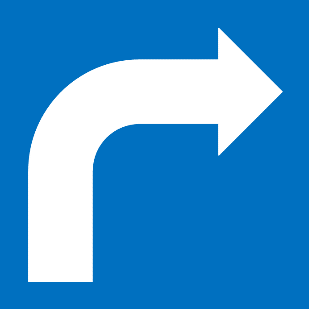
np.sum – tömb eleminek összegét számolja

**C:\Users\Noel\Documents\gepi\wrr.PNG16.)** Megadjuk a szegmensek értékét.

tuple – egy kollekció ,ami rendezve van és nem megváltoztatható

**17.)** Döntéshozatal, eredmény kiírása a képernyőre.

**18.)** Kilépés a ’q’ gomb megnyomásával.

****

**Források:**

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html>

<https://stackoverflow.com/questions/43111029/how-to-find-the-average-colour-of-an-image-in-python-with-opencv#43111221>

<https://github.com/nikgens/TankRobotProject/tree/master/signRecognition>

<https://www.tankonyvtar.hu/hu/tartalom/tkt/angol-magyar/ch22.html>

<https://docs.scipy.org/>

<https://www.w3schools.com/python/>