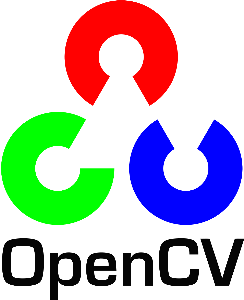
**Dokumentáció**

Az alábbi dokumentáció egy olyan programról lesz, ami egy egyszerű algoritmus alapján, képes felismerni korlátozott számú, előre meghatározott közlekedési táblát. Az algoritmus alapja a kör felismerés, zónákra osztás és a zónák domináns színének detektálása.

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

* Kamera
* ****Thonny IDE
* OpenCV library
* ****Numpy
* Python

**Felhasznált külső könyvtárak:**

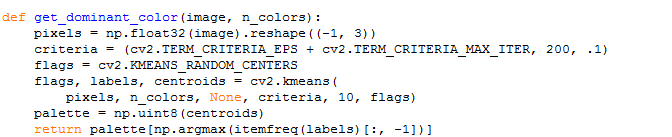
* cv2:OpenCV elérés
* itemfreq : 2-D frekvencia-táblázat. Az 1. oszlop rendezett, egyedi értékeket tartalmaz a 2. oszlopban.
* Numpy : kiegészítő csomag a Python programozási nyelvhez magas szintű matematikai függvénykönyvtárral.

**Az algoritmus:**

**Először** is elvégezzük a modul hívást.

**C:\Users\Noel\Desktop\signs\1.PNG**

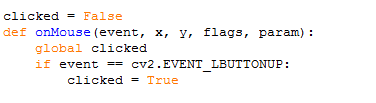
**C:\Users\Noel\Desktop\signs\sdw.PNGMásodik** lépésben definiáljuk a függvényt. Ennek a célja a színek felismerése és arányuk meghatározása. Végül pedig eldönti ,hogy a képen melyik szín van a legtöbbet használva.



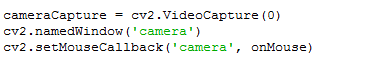
C:\Users\Noel\Desktop\signs\sww.PNG : Beállítjuk a program iterációit ,ami MAX-nál áll meg végül.

C:\Users\Noel\Desktop\signs\ss.PNG : kezdeti pontok meghatározása.

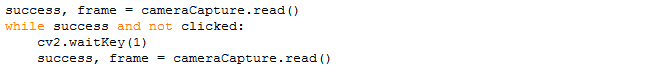
**Harmadik** lépésben az egér kattintására szeretnénk elindítani a kamerát, amit a következő módon tudunk megvalósitani:



Az onMouse függvény a bal egérgomb felengedése után megjegyzi a kurzor x és y koordinátáját.



setMouseCallback pedig megjeleníti a kamera közvetítését az egér pillanatnyi helyzeténél a „kattintás” után.

A **negyedik** lépésben beállítjuk a kamera képének frissítési gyakoriságát.

A cv2.waitKey(1) 12 ms-es frissitést eredményez.

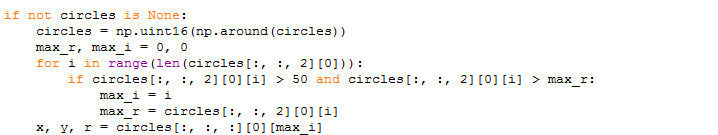
C:\Users\Noel\Desktop\signs\ssww.PNGA nagyobb sebesség elérése érdekében a színeket fekete-fehérré alakítjuk a következő sorral:

Ezután elhomályosítjuk a képet az esetleges háttérben felbukkanó kör alakú objektumok érzékelésének elkerülése céljából :

C:\Users\Noel\Desktop\signs\hgfdw.PNG

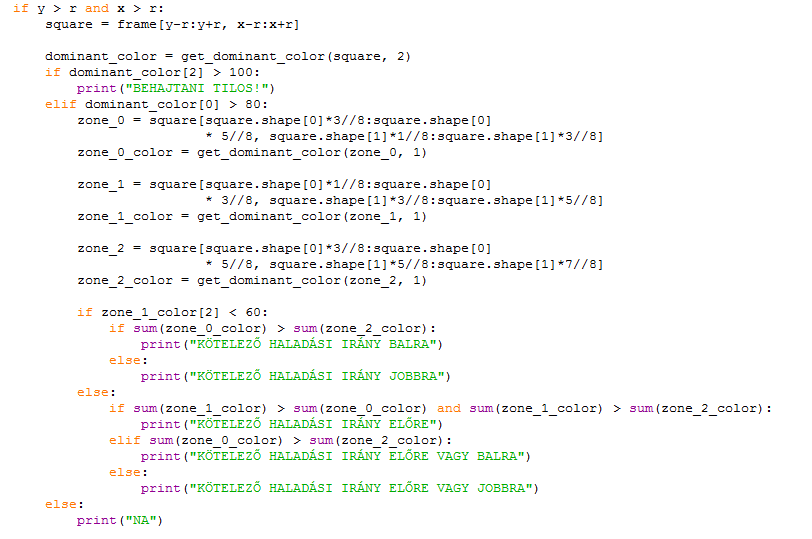
A **ötödik** lépésben egy beépített OC:\Users\Noel\Desktop\signs\wert.PNGpenCV algoritmust hívunk meg.

Az algoritmus fontosabb paraméterei a következők : a minDist(jelen esetben az 50-es szám) azt a minimum távolságot jelzi,ami két különböző érzékelt kör középpontja közt kell,hogy legyen. Ezzel elkerüljük,hogy egyszerre több kör legyen egy helyen. Viszont ha túl nagyra állítjuk, akkor néhány kör kimaradhat. A param1 növelésével emeljük az érzékelhető körök számát,a param2 növelésével pedig több hamis kört ejtünk.

**Hatodik** lépés a kamera érzékelési távolságának beállítása

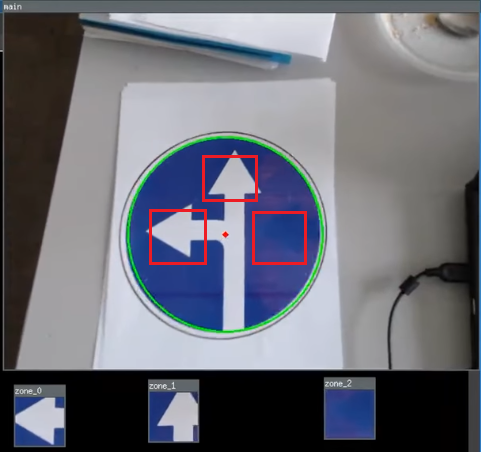
Az utolsó sor az előbbi kódrészletben ellenőrzi a kör távolságát. Ha a kör a határon kívül esik, akkor kivágunk egy négyzetet a körrel a középpontban, ezzel elkerüljük a program összeomlását.

A **hetedik** lépés a táblák felismerése.



Mivel kevés táblával dolgoztunk könnyű dolgunk van a felismeréssel. Ha az érzékelt tábla piros, akkor kiírathatjuk ,hogy ez egy „Behajtani tilos” tábla,mivel csak ebben az esetben domináns ez a szín.

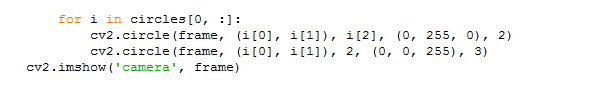
Ha az érzékelt szín más,mint piros akkor a program a kört három zónára osztja. A zónák az érzékelt körben előre meghatározott területeken helyezkednek el, az alábbi módon:



Amikor csak balra vagy jobbra haladunk a felső 1-es zóna domináns színe a kék, tehát a maradék két zónát vizsgálom. Ha a 0-ás zóna domináns színe fehér,akkor kiíratjuk,hogy a tábla a „kötelező haladási irány balra” ellenkező esetben pedig kiírathatjuk,hogy a táblánk „kötelező haladási irány jobbra”. Ha az 1-es zónában fehér a domináns szín, szintén tovább vizsgálom a maradék zónákat. Itt három lehetséges eset van:

1. Mindkét zónát a kék dominálja „Kötelező haladási irány előre”
2. A 0-ás zónát fehér, a 2-es zónát kék „Kötelező haladási irány előre vagy balra”
3. A 0-ás zónát kék, a 2-es zónát fehér dominálja „Kötelező haladási irány előre vagy jobbra”

**Nyolcadik** lépésben megjelenítjük az érzékelt kört egy külön ablakban.



Végül pedig bezárjuk az összes ablakot,és befejezzük a kamera rögzítését.C:\Users\Noel\Desktop\signs\eeea.PNG

**Források**:

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

<https://www.pyimagesearch.com/2016/02/08/opencv-shape-detection/>

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

<https://github.com/vamsiramakrishnan/TrafficSignRecognition>

<https://github.com/foxpy/road-sign-recognition>