**Közúti jelzőtábla felismerés**

A “RoadSignDetection.py” egy futtatható, python programnyelveben írt rövid program (), mely teljes kapacitásában 42 különböző féle táblát képes felismerni.

A “sample”, “reference.zip”, valamint “RoadSignDetection.py” és “RoadSignDetection.pyc” állományokat ajánlott egy külön mappába helyezni leöltés és kicsomagolás után, a problémamentes futtatás érdekében.  
Az futtatáskor meg kell adnunk a felsimerni kívánt kép (ha ez nem a projektmappában van, akkor teljes elérési úttal együtt) almappáját és nevét a következő formában: ‘(elérésiút/)almappa/képneve’ . Ez után Enter bilentyűt nyomva elindul a feldolgozás.

Mivel a program a “reference” mappában lévő képekkel hasonlítja össze az adott képet, így az inputként használni kívánt képet a “sample” nevű mappából érdemes https://opencv-python-tutroals.readthedocs.io/en/latest/py\_tutorials/py\_feature2d/py\_feature\_homography/py\_feature\_homography.htmlkiválasztani.

Az információgyűjtés során szerzett tapasztalataim szerint a problémát legtöbbször, valamint éles helyzetben Deep Learninggel szokták megoldani, azonban mivel a félévben tanultak nem ehhez kapcsolódtak és a jelenlegi képességeimet is meghaladná ez a feladat (vagyis határidőn belül biztosan), így egy egyszerűbb, de kevésbé hatékony megoldást tudtam megvalósítani, amely az azonosítani kívánt kép és az adathalmaz többi képét hasonlítja össze, majd megfelelő számú egyezés után vissza adja a kapott eredményt stringként.

Első lépésként előre definiálom egy listába a táblák neveit a mappák sorrendjében, majd pedig egy másik listában a mappaneveket, melyeket string változóként adok hozzá az elérési útvonalhoz.

A main függvényt lényegében egy while ciklus adja, mely az adathalmaz minden almappájában lévő képet egyesével végig “néz” mindaddig, míg az épp vizsgált mappában lévő képek és az input kép közt nincs átlagosan legalább 10 egyező pont.

Ebben egy nested for ciklus a soron lévő képet össze hasonlítja az inputként definiált képpel. Itt a soronlévő kép beolvasásra és átméretezésre kerül (300px \* 300px), létre jönnek a key pointok és leírások mindkét képhez, majd pedig egy újabb for cikluson belül bővül a jó találatok listája, ha a két pont közti távolság elég alacsony.

A megfelelő egyezések érdekében forgatásra nem volt szükség, a grayscale pedig nem segített több egyezés találásában sokadik tesztelés után sem, így a szükséges számítási kapacitás csökkentése érdekében ezt kihagytam.

Irodalomjegyzék:

<https://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/published-archive.html>

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html>

https://opencv-python-tutroals.readthedocs.io/en/latest/py\_tutorials/py\_feature2d/py\_feature\_homography/py\_feature\_homography.html