**Road Sign Detection**

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

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.

A használt adatbázisom az Institute für neuroinformatik weboldaláról származik, a German Traffic Sign Recognition Benchmark képeit használtam.

A kód első “signs” változója egy 43 stringet tartalmazó lista. Ez tartalmazza az összes tábla nevét abban a sorrendben, ahogy a mappaszerkezetben azok egymás után következnek. A következő, szintén 43 stringből álló “pathVariables” nevű lista tartalmazza a a mappák neveit, melyekben a képek vannak.

A következő sorokban a program bekéri a felhasználótól a vizsgálni kívánt kép elérési útvonalát és nevét. Miután megtörtént az adatbevitel, a program a képet újraméretezi 300 x 300 pixeles méretűre a könnyebb munka érdekében.

Ezt követi 3 integer típúsú változó meghatározása. Az első, “filec” változó fogja számolni hány darab képet vizsgáltunk meg a futás adott pontjáig az aktuálisan vizsgált mappában. A következő “percentage” nevű változó valójában nem százalékértéket tárol és nem is azt ad vissza. Ez fogja számolni a futás adott pontjáig a program által az adott mappában a képeken megtalált összes egyező pontot. Ezt követi az “i” változó, ami az eddig megvizsgált mappák számát fogja számon tartani.

A kód egy while ciklussal folytatódik, mely addig tart, míg az adott mappában található képeken talált egyezések elosztva a mappában vizsgált összes kép számának tizedével eredménye alacsonyabb, mint 10, vagy át nem vizsgáltuk az összes rendelkezésre álló mappát, melyet a while ciklus vége előtt az “i” változóval lehet meghatározni.

A cikluson belül először resetelem a “percentage” és a “filec” értékét és létrehozok egy új segédváltozót, ami szintén ciklusonként resetelődni fog. Ez után a “pattern” változóban meghatározom az elérési útvonalat a mintaképekhez.

Itt a while cikluson belül van egy nested for cilus, mely az össes mappában talált képet beolvassa, majd 300 x 300 pixelesre méretezi a könyebb munka érdekében. Ez után “sift”-tel megahtározom a keypointokat és descriptorokat. Eredetileg “orb”-ot szerettem volna használni, de ez az openCV általam használt 3.2-es verziójában már nem érhető el, így ezt találtam a legjobb megoldásnak.

A következőkben FlannBasedMatcher-rel keresem a legközelebbi szomszédokat, majd az aktuális egyezéseket egy listába írom. Ez után meghatározom az egyezések számát és ezt hozzá adom a “percentage” korábbi értékéhez, ezzel meghatározva az adott mapában található képeken talált addigi összes egyezést.

A “filec”-t léptetem egyet, mivel egy kép átvizsgálása megtörtént. Ha ez a szám eléri a 40-et az algoritmus kilép a for ciklusból es folytatja a mappák átfésülését.

A ciklusból kilépés után ellenőrzöm az addig átvizsgált mappák számát. Ha ez eléri a kritikus értéket (jelen esetben 19, mert annyi mappa fért el githubon), az algoritmus a while ciklust is elhagyja, és kiírja az kapott eredményt.

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.

**Tesztelés**

A tesztelések során egy problémába futottam bele, ami abból állt, hogy a jellemzően 30 x 30 pixelnél kisebb, nagyon sötét képeken nem talált a program elég összehasnolítható pontot. Ezeknek egy jelentős részét szabad szemmel magam sem nagyon tudtam felismerni, csupán tippjeim voltak a körvonal alapján, így eltávolításra kerültek. Ezek eltávolítása után (mind a reference, mind a sample mappából szelektáltam a hasonlókat), a jelenlegi 75 képből álló datasettel a program megközelítőleg 85%-os pontossággal fut le.

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>

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

<http://benchmark.ini.rub.de/?section=gtsrb&subsection=news>