Gépilátás beadandó

2019/2020/2

Feladat: Rubik kocka állapotának felismerése

Készítette: Gvárdián Levente

Neptunkód:U7A44W

Bevezetés

Rubik kocka felismerő alkalmazást választottam féléves feladatnak. Ez az alkalmazás 6 fényképet olvas be, amin a rubik kocka minden egyes oldala külön-külön található meg, és ezután átkonvertálja a képet HSV színtérbe, ez alapján készít maszkot, és minden maszkra külön kontúrkereséssel, majd megrajzolással készít az eredetileg beadott képre kontúrokat, azok köré virtuálisan egy kör alakú kontúrt képez, és ennek a körnek a középpontját olvassa be. Ekkor egy koordinátát kapunk, illetve egyből a színt is elmenti egy kétdimenziós tömbbe, majd ezután ezt az x,y,szín koordinátát sorbarendezzük y majd x szerint, és így megkapjuk a kockákat sorrendben balról jobbra haladva. Ezt már csak kiíratással a végén kiírattam a konzolba.

Elméleti háttér

Elméleti háttere a feladatomnak, hogy először is ugye a beolvasott képet RGB, színrendszerben kapom meg. Ekkor az egyeztetett első konzultáció alapján, másik színrendszert kerestem, és kis internetes kutatás után HSV színrendszert találtam a legmegfelelőbbnek. Amint ez sikerült, és átkonvertálta a képet a program a számunkra megfelelő színrendszerbe, ekkor következett a maszkolás problémája. Hiszen szükséges tudni a kívánt szín maszkolásához a szín HSV-ben megadott határértékeit, amit lower, illetve upper előtaggal adtam meg. Ehhez egy kis interneten talált program általi segítséget kellett alkalmaznom, hogy ezt a lower,és upper határokat megtaláljam minden színhez HSV szerint. Ezt követően jött a következő probléma, hogy hogyan határozzam meg hol van melyik kocka, illetve milyen színű az a kocka. Ehhez ismételten a konzultáció alapján használtam a kontúrozást. A koordinátákat pedig úgy kapom meg, hogy ugye a kontúrrajzolás az abban merül ki, hogy a képen látható 9 kockára ugye a maszk alapján rajzolja a négyzet kontúrokat, ami köré egy függvény segítségével a lehető legkisebb területű kört “rajzoltattam” virtuálisan, amelynek megnézem a középpontját, és az a középpont visszaadott értéke lesz az x és y koordináta. Ehhez tömböt használtam, ami egy 9x3-mas mátrix, hiszen mind a 9 kockának van x,y koordinátája, és egy színe, amit szintén eltároltam az alapján, hogy a maszkolás for ciklusa hányszor futott le, hiszen ahányszor lefutott már a for ciklus, annyiadik szín kerül maszkolásra, ami alapján a kontúrt rajzolja a program. Amint ez megvan, és ezeket eltudom tárolni, akkor szükségem van egy sorrendbe rendező algoritmusra, amely a tömb elemeit az x,y koordináta alapján rendezi el, úgy hogy a kockákat balról jobbra kapjuk meg. Ha ezt is megkaptuk, akkor már csak kiíratjuk sorrendben a színeket, és kész a program.

Megvalósítás terve és kivitelezése

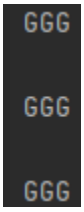
Megvalósítás első része a kép beolvasása. Először is ugye szükségem volt a numpy és opencv plugin segítségére. Ezeket beimportáltam a program elején. A képeket beolvastatom a program elején, majd HSV-be konvertálom a cv2.cvtColor segítségével. Ha ezzel kész van, akkor következik egy forciklus ami azt fogja jelenteni, hogy mind a hat maszkra (mivel hat szín van a rubikkockán, és színenként szeretnénk maszkolni) a cv2.inRange segítségével maskot készítünk, ami után egyből következik a kontúrokkal való foglalkozásnak a része. Itt elsőzöris a kontúrokat megkeresi a program, és szépen végig rajzolja, de egy if segítségével a kis területű kontúrokat ki tudjuk szűrni, tehát aminek területe (pixelben) nagyobb mint 10000, akkor megrajzolja nekünk a kontúrt. Eköré a kontúr köré készítünk egy kör alakú kontúr, ami pontosan ráilleszkedik a rajzolt kontúr négyzetre, és ennek a körnek a közepe fogja nekünk megadni a koordinátákat, tehát a kör középpontja a alapján határozom meg a kockákat. Ez azért is fontos, mivel különben nem tudnánk a végén a kiíratásnál, hogy melyik kockát szeretnénk kiíratni. Ezt követően egy kétdimenziós tömbbe tárolom el, aminek 9 sora és 3 oszlopa van, hiszen minden kockának van egy koordiánátája, amit a kör alapján határoztunk meg, illetve minden kockának van egy színe, amit egy "j" változó segítségével határozok meg. Ehhez szükséges tudni, hogy hányadik for ciklusban járunk a maszkolásnak, így a "j" változót minden maszk forciklus végén növeltem eggyel. Tehát ezalapján tudjuk, mivel a színek így vannak sorrendben: zöld, kék, piros, sárga, narancs, fehér. Hiszen ezeket én adtam meg a lowup tömbnél, aminek a célja ugye a színek kiszűrése HSV segítségével, ezt használja a maszk is. Szóval ez is eltárolódik a cszintomb-ben tehát 9x3 mátrix szükséges, hogy mind a megtalált kilenc kockánknak x,y koordinátáját, és színének értékét(ami egy int, de mivel bekerül a tömbbe ezáltal float típusú lesz a koordinátákkal együtt). Ezt követően jön a segéd tömb bevezetése, ami ahhoz kell, hogy a cszintomb-ből külön szedjem a float típusokat külön értékekké, hiszen a cszintomb nem teljesen 9x3-mas mátrix, hanem csak egy float tömb ami 1x27, hiszen az np.append segítségével adtam hozzá a dolgokat. Visszatérve, ebből egy mátrix lesz a segéd tömb, azaz a programomban segedt alapján. Viszont ez így nem maradhat, hiszen koordináták alapján szeretnénk ezt sort-olni, tehát rendezni, és ezt is úgy, hogy először az y, majd x koordináták alapján. Így megkapva mind a 9 kockát balról jobbra haladva, fentről lefelé. Ezt követően már csak annyi a feladat, hogy ezt konzolba kiírjuk, tehát megadja, hogy melyik szín, hol van. Ezt leegyszerűsítettem: G,B,R,Y,O,W betűrkre, hiszen így is tökéletesen érzékelhető, hogy a programnak mi az eredménye. Sajnos nem sikerült megoldani, hogy mind a 6 képet amit beadunk neki, azt automatikusan magától egyesével végig menjen rajtuk, de még utólag próbálkozom vele, hátha azt is sikerül, és akkor majd feltöltöm külön kódfájlként.

Tesztelés működő képekkel

Tesztelésem során megadtam a következő képeket:

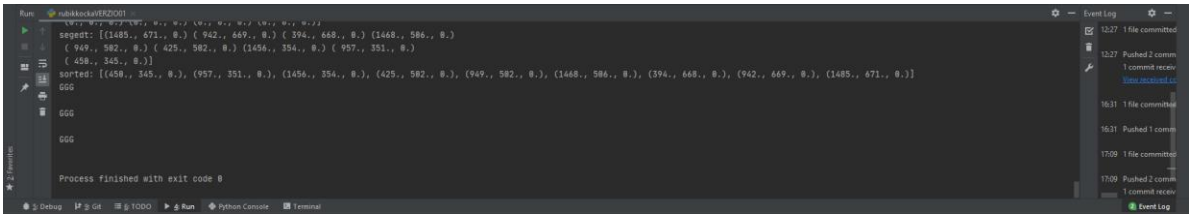
1.ábra: zold.jpg, 2.ábra: kek.jpg, 3.ábra: piros.jpg, 4.ábra: sarga.jpg, 5.ábra: narancs.jpg, 6.ábra: feher.jpg

Eredménye a tesztelésnek:



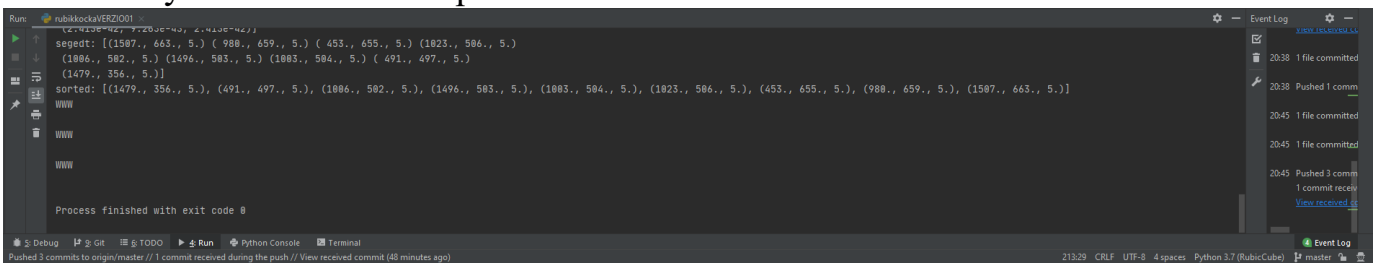
7.ábra: zolderedmenye.png

Tesztelésemhez szükségem volt erre a 6 képre, amelyet én csináltam itthon a saját rubikkockámról. Nem kevertem össze, mert először is úgy akartam a programot működtetni, hogy egy oldalon minden kocka ugyan olyan színű, és felismeri-e rendesen a program. Tesztelésem eredménye a 7.ábra, amelyen láthatjuk, hogy ha zöld színű oldalát olvastatom be a programmal, akkor eredményként kiírja helyesen. Ezt természetesen végig csinálva minden oldallal jó eredményeket kapok, de feleslegesnek tartottam beszúrni mindegyiknek eredményét, inkább itt egy képernyőkép a tesztelésről, amint a kódot lefuttatom, és konzolban látszódik a végeredmény:

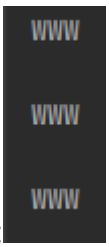


Tesztelés zavaró tényezős képpel

Zavaró tényezőkkel ellátott kép tesztelése:



Ahogy az látszódik itt is:



Hiszen W a fehéret jelenti, és ez az első zavaró tényezős kép eredménye, amin pirosat kellett volna érzékelnie.

Felhasználói leírás

Először is szükségünk van megadni az imread részeknél a 6 képet, amivel beolvastatjuk a programba. Ezt a 4.sortól a 11.sorig tehetjük meg. A zárójeles részbe kell beírni a fáj pontos nevét és ponttal elválasztva a kiterjesztését. Ha végeztünk, akkor a 186. Sorban kell átváltoztatni a cvt.Color(ide kell beírni a változó nevét, amelyik képet szeretnénk tesztelni), majd ugyan ezt a 196.sorban is a cv2.rectangle(ide kell beírni a változót. Ha ezzel végeztünk, akkor csak elindítjuk a programot és kidobja erre a képre az eredményt, majd ezt ismételjük az összes képpel, amit tesztelni szeretnénk.

Felhasznált program:

Ide inkább egy kódot szeretnék megadni, amit feltöltöttem a githubra, ami a hsv detektálásában segített, hogy a színeket jól tudja maszkolni a program, és ennek a fájlnak a neve:hsv_detection.py

Természetesen ez nem az én munkám, de nem találtam meg sajnos a linket, amelyről letöltöttem.