Gépi látás

Rubik kocka színeinek az érzékelése

Földes Kálmán Viktor

MOTRPB

2020.05.30

Tartalomjegyzék

1 A megoldandó feladat kifejtése 2
2 Megoldáshoz szükséges elméleti háttér 3
3 Megoldás terve és kivitelezés 4
3.1 Tervezés 4
3.2 Kivitelezés 4
4 Tesztelés 8
5 Felhasználói útmutató 13
6 Felhasznált irodalom 14

1 A megoldandó feladat kifejtése

Azt a feladatot kaptam, hogy olvassam be a Rubik kocka oldalain található színeket és tároljam el mind a 6 oldalt. Ezeket az oldalakt a program legvégén kiírom a rajtuk található színekkel. A programot python programozási nyelven fogom írni. használni fogom a PIL, numpy és cv2 könyvtárakat. Az elején bemutatom az elméleti hátteret, hogy mit csináltam a képpel és a függvény hogyan csinálta és hogy hogyan számoltam ki, amit saját magam írtam. Utána a tervezés részben leírom ,hogyan gondoltam a megvalósítást nagy vonalakban. Bemutatom a kódot. Az 5. részben bemutatom a program eredményeit, amit működés közben csinál a program. Majd legvégül mutatok működő és nem működő példákat, hogy lehessen látni mikor jó a program és milyen esetben nem alkalmas a kép. Legvégül a felhasználói útmutatóval leírom egy felhasználónak hogyan tudja működtetni a programot és mit lehet állítani benne, ha esetleg nem a kívánt végeredményt kapta pedig majdnem jó a kép.

2 Megoldáshoz szükséges elméleti háttér

A kép világosabbá tételéhez PIL könyvtárat használtam ez egyszerűen csak az RGB kódot felszorozza egy bizonyos értékkel, amit megadok. De nem hagyja, hogy 255 felé menjen az érték.

A fekete szín felismeréséhez a képet Cv2 könyvtárral átalakítottam egy másik színtartományba. HSV-be BGR-ről. Erről a színtartományról azt kell tudni, hogy A színárnyalata 0 tól 179-ig tart, a telitettség 0tól 255-ig tart, értéktartomány a pedig 0-tól 255-ig. [1]

Az alakzat felismerésénél szűrkeárnyalatossá alakítja a képet, ami egy súlyozott átlagolás a 3 szín (BGR) átlaga alapján állapítja egy képpontban, az összes képpontnál. Ez adja meg az intenzitását a kreált képen a fekete pontoknak. Ezután a threshold binárissá alakítja a fehéret (1) és a feketét (0). Utána a kontúr kereső megkeresi a kontúrok pontjait és eltárolja. Ezek függőleges, vízszintes vagy átlós pontok, majd hierarchia szerint rendezi. A négyszöget úgy ismeri fel, hogy csak 4 irányváltása volt, szóval 4 koordinátát nyerünk csak ki [2]

Az oldal arányok kiszámítása elég egyszerűen magasság/hossz, ezt arra használom, hogy megállapítsam, hogy mennyire aránytalan a négyzet.

1) Ábra

A számolásom illusztráltam fent látható képen. Az A+(B-A)/C képlettel számoltam. Ebben az A lesz az egyes pont utána a B a kettes pont koordinátája ebből kiszámolom a hármas pontot. Az osztó a C ilyenkor kettő értéket kap. Ezután kiszámolja a második felezőpontot 4. és az 5. pontból, Ezekután kiszámolja az 1/2 , 1/8 és a 7/8 részét a piros vonalnak Ebből megkapva a 3 színt ,majd Utána újrakezdi az egészet a bal és jobb oldal nyolcadat számolja ki és a felső részen csinálja ugyan ez majd legvégül az alsó részt is kiszámolja

A program átlagot is számol A kapott pontokat körbejárja a program egy adott nagyság alapján és úgy átlagolja a színeket. Ha 1 nagyságú sugarat adunk meg akkor az eredmény képpont körül levő 8 másik pontot hozzáadja és elosztja 9-cel. így kapva meg az átlagot.

3 Megoldás terve és kivitelezés

3.1 Tervezés

A képen a kocka egy oldala lesz látható. Ezt az egy oldalt fogom azonosítani. Ennek az lesz a hátránya, hogy a kép elkészítésekor tudnom kell melyik oldal merre van. Mivel ezt a program nem tudja megmondani majd. A kocka detektálásához a fekete színt fogom kimaszkolni. Ezért a képeket nagyon fényes háttérrel, lehetőleg teljesen fehér, árnyékmentes környezettel fogom megcsinálni. Miután a fekete részt felismertem, megkeresem a legnagyobb négyszöget a képen és elmentem a koordinátáit. Ezeknek a koordinátákat arra fogom használni, hogy kiszámoljam hol milyen szín található. Ezeket RGB kóddal fogom felismerni. Majd a végén eltárolom egy tömbben és kiíratom. A színek betű formájában tárolódnak el.

3.2 Kivitelezés

Első lépésben meghívom a numpy könyvtárat a tömbök létrehozásához kell. Másodjára a cv2-t ami a képek konvertálására, éldetektálásra kell. Harmadjára a Pillow könyvtárat, ami a fényerősséget állítja be a képnél.

1. Kódrészlet

```
import numpy as np
import cv2
from PIL import Image, ImageEnhance
```

A 2. kódrészletben létrehozok egy üres tömböt, ami hatszor tartalmaz 3x3-as tömböket. Ebben fogom tárolni a 6 oldal színeit. A színeket a Cube végeredménytömbben tárolom majd, ahhoz, hogy tudjam melyik oldalhoz tartozzon melyik szín már előre feltöltöttem a közepét-a megfelelő betűvel.

2. Kódrészlet

```
Cube = np.zeros((6, 3, 3),dtype='U')


for i in range(6):
    if i==0:
        Betu="R"
    if i==1:
        Betu="W"
    if i==2:
        Betu="O"
    if i==3:
        Betu="Y"
    if i==4:
        Betu="G"
    if i==5:
        Betu="B"
    Cube[i][1][1]=Betu
```

Az M ciklussal fogom jelölni a kép nevét. A képeket 0-tól számozom 5-ig.De akár több mintát is lehet bele rakni vagy kevesebbet. utána im változóba behívom a vizsgált képet és fényesebbé teszem. A képen alapból 1.2-es szorzó van, de ezt lehet emelni, ha árnyékosabb a kép. Majd lementem egy vilagos.jpg nevű képben, hogy nyomon lehessen követni mit csinált vele.

3. Kódrészlet

```
for M in range(6):


    im = Image.open(str(M)+".jpg")
    enhancer = ImageEnhance.Brightness(im)
    enhanced_im = enhancer.enhance(1.2)
    enhanced_im.save("vilagos.jpg")
```

Beolvasom a felvilágosított képet és hsv-re alakítva a képet megkeresem a feketét a képen a látható tartományon belül. Majd elmentem az eredményt.

4. Kódrészlet

```
img2 = cv2.imread("vilagos.jpg")
hsv = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)
lower_range = np.array([0,0,0])
upper_range = np.array([245,180,125])
mask = cv2.inRange(hsv, lower_range, upper_range)
cv2.imwrite("smask.png",mask))
```

Ebben a részben a képet átalakítja majd kontúrt rak rá. Utána detektálja a négyszögeket, de csak azzal a feltétellel hogyha az eddig legnagyobb oldallal és hosszal rendelkezik és legalább 1/2 vagy 3/2 arányban van az oldalhossz és a magasság. Erre azért van szükség mert a legnagyobb négyzetet keresem és téglalap legyen legalább, nehogy valami hosszú csík bezavarjon és azt higgye a legnagyobb négyzetnek.

5. Kódrészlet

```
w2=0
h2=0
img = cv2.imread('smask.png')
imgGrey = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
_, thrash = cv2.threshold(imgGrey, 240, 255, cv2.THRESH_BINARY)
contours, _ = cv2.findContours(thrash, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)

for contour in contours:
    approx = cv2.approxPolyDP(contour, 0.01* cv2.arcLength(contour, True), True)
    cv2.drawContours(img, [approx], 0, (169, 100, 100), 5)
    x = approx.ravel()[0]
    y = approx.ravel()[1] - 5
    if len(approx) == 4:
        x1 ,y1, w, h = cv2.boundingRect(approx)
        aspectRatio = float(w)/h
        if w > w2 and h>h2 and aspectRatio >= 0.50 and aspectRatio <= 1.5 :
            w2=w
            h2=h
            t=approx
```

A Cn tömbben fogom tárolni a felismert színeket sorban. Ebben 9 elem lesz. Az sz tömb pedig 6 értéket fog tárolni. az első kettő a fenti koordináta a másik kettő pedig az alsó, az utolsó kettőben pedig a kettő között lévő értéket tárolom és majd a kiszámolt színek helyét.

6. Kódrészlet

```
CN=[]
sz=[1,2,3,4,5,6]
```

Utána for ciklusban kiszámolom az értékeket az a+(b-a)/c képlettel. A c értéke a változtatásával számolom ki az 1/8 és a 7/8 részét a kockának.

Itt az eredeti képet használom nem a világosítottot mert az módosított értékkel lenne. A kapott pontot, amit kiszámoltam azt körbejárom és átlagolom végül az AVG változóba mentem. Ezt az színátlagot, a color változóban eltárolom. Colorban a BGR kódot fogom iffekkel megnézni milyen színű. Majd angol megfelelőját belerakom a CN-be. Az intervallum növekszik, ha nem talál meg egy színt. de ha már nagyobb mint 150 és nem találja akkor kilép. Bár ez akkora intervallum, hogy biztosan megtalálja. De volt rá példa, hogy 100 kevés volt.

7. Kódrészlet

```
        color=AVG
        interv=10
        while interv<150:
            interv=interv+5
            #piros
            if color[2]>=150-interv and color[2]<=150+interv and color[1]>=45-interv and color[1]<=45+interv and color[0]>=45-interv and color[0]<=45+interv :


                CN.append("R")
                break
            #sarga
            if color[2]>=170-interv and color[2]<=170+interv and color[1]>=180-interv and color[1]<=180+interv and color[0]>=60-interv and color[0]<=60+interv :


                CN.append("Y")
                break
…
```

A legvégén pedig Cube-tömbbe eltárolom a 9 színt, amit talált, betű formában. Példa egy oldal eredményére:

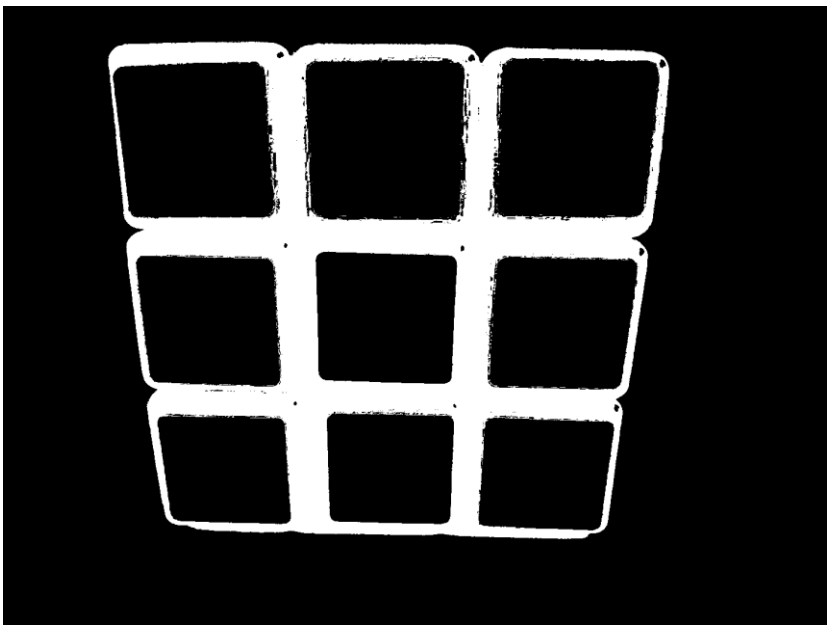
```
[[['W' 'W' 'W']
 ['B' 'R' 'Y']
 ['W''O''Y']]
```

De persze üres oldalakat is kiírja majd mellé eredménynek.

4 Tesztelés

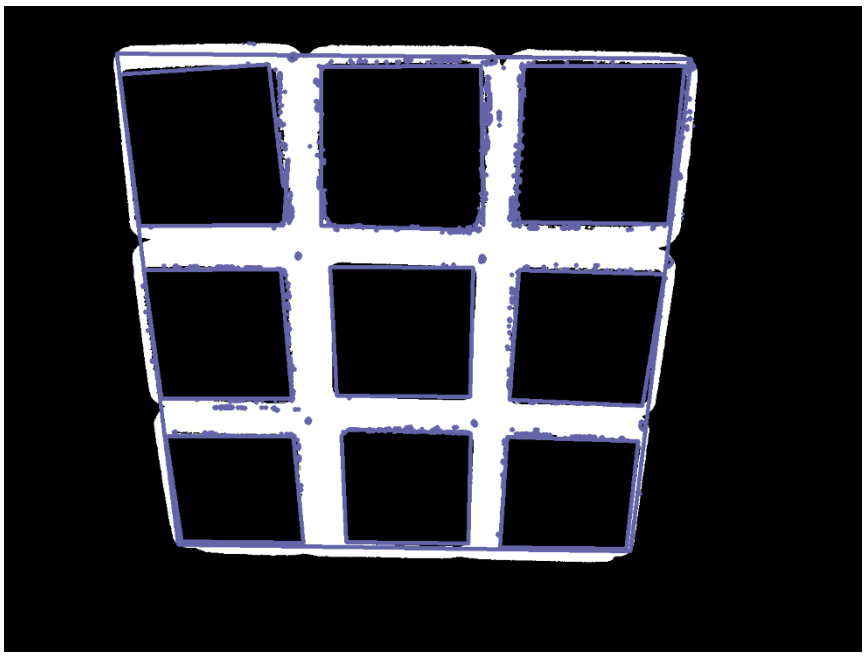
A mellékelt mappában van egy ”A kocka 6 oldala” nevű mappa amiben a kocka 6 oldalát lefényképeztem megfelelő körülmények között. Szóval fehér a háttér nincsen árnyék és jól látszik a kocka adott oldala.
Ebből a mappából most a 0.jpg nevű képpel fogok tesztelni.

Ez a fekete részek fehér kiemeléssel



2) Ábra

Ezután a maszkunkra lila kontúrt rakott. Mint látható sikeresen kijelölte a kockát bár nem túl pontosan ezért a számítások pontatlanok lesznek. A kisebb kockákat nem baj, hogy jelölve vannak mivel a legnagyobbat fogjuk lekérni



3) Ábra

A következő koordinátákat kapjuk

[[[189 79]] ez a bal felső ,

[[292 900]] a bal alsó,

[[1051 911]] jobb alsó,

[[1153 88]]] jobb felső sarok.

Eredményű kaptuk

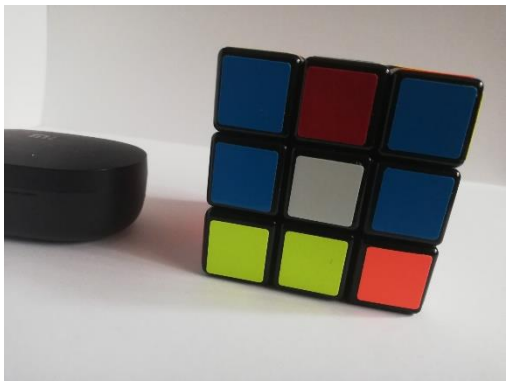
[['O' 'Y' 'Y']

['B' 'W' 'B']

['B' 'R' 'B']]

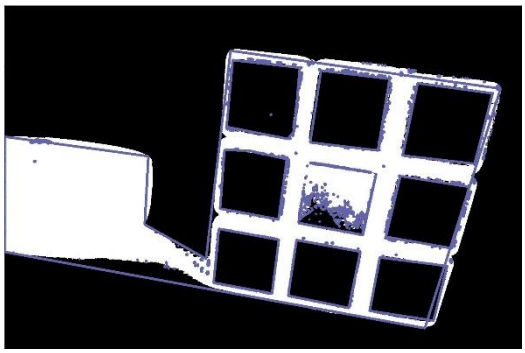
Ha megnézzük a képet reális eredményt kaptunk. A hat képet tesztelve mindegyikre jó eredményt kaptam mivel megfelelő környezetben készült a képek.

Ezekután teszteltem olyan képekre, amik hibát okozhatnak. Elsőnek olyat, amin van fekete tárgy és árnyék is



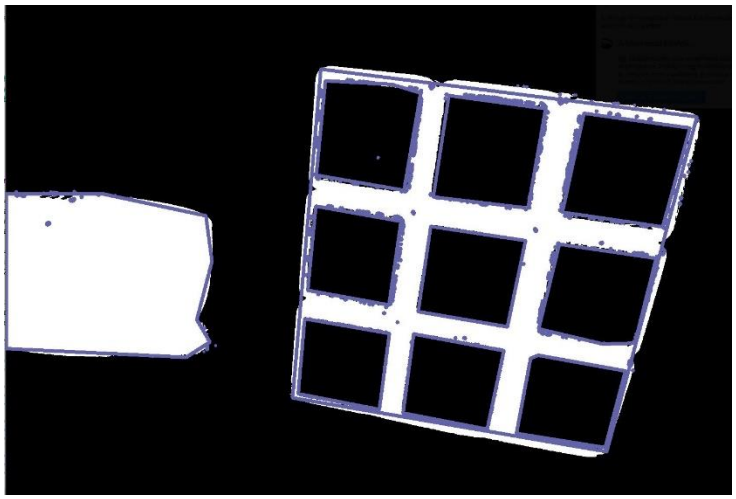
4) Ábra

Ha a fényerő nincs növelve akkor az árnyék problémát okoz ezért tettem be a fényerő növelést.



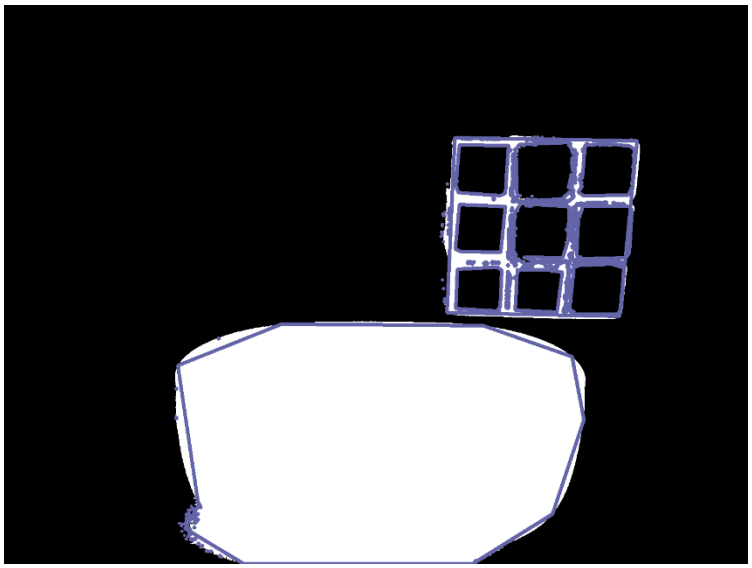
5) Ábra

De amint a fényerőt beállítottam 1,2-re a programon belül az árnyék már nem lett probléma ezen a képen mivel halvány árnyék volt. A fekete objektum se zavarta meg mivel nem négyzetalakú és még kisebb is



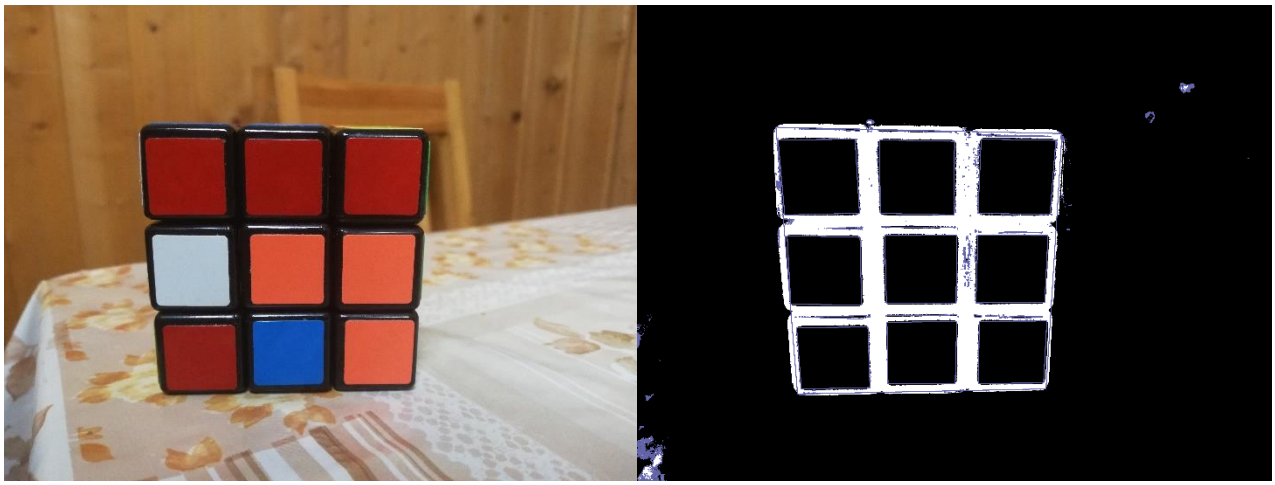
6) Ábra

A 3.jpg képen tesztelve is működik az éldetektálás hiába van egy nagyobb fekete tárgy előtte nem azt nézte mert nem volt négyzet.



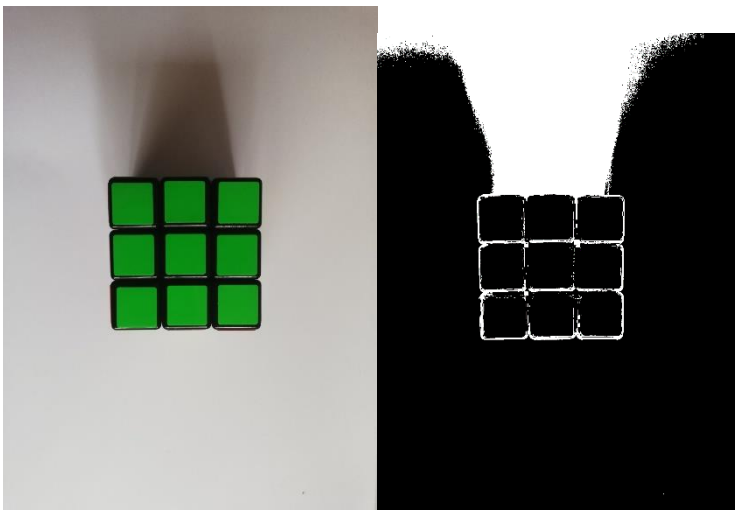
7) Ábra

A következő 11.jpg nevű kép nem egyszínű háttérrel rendelkezik, mégis világos és árnyék nélküli ezért feltudja ismerni a nagy négyszöget. Mint a következő kép is mutatja. A színfelismerési eredmények is jók lette.



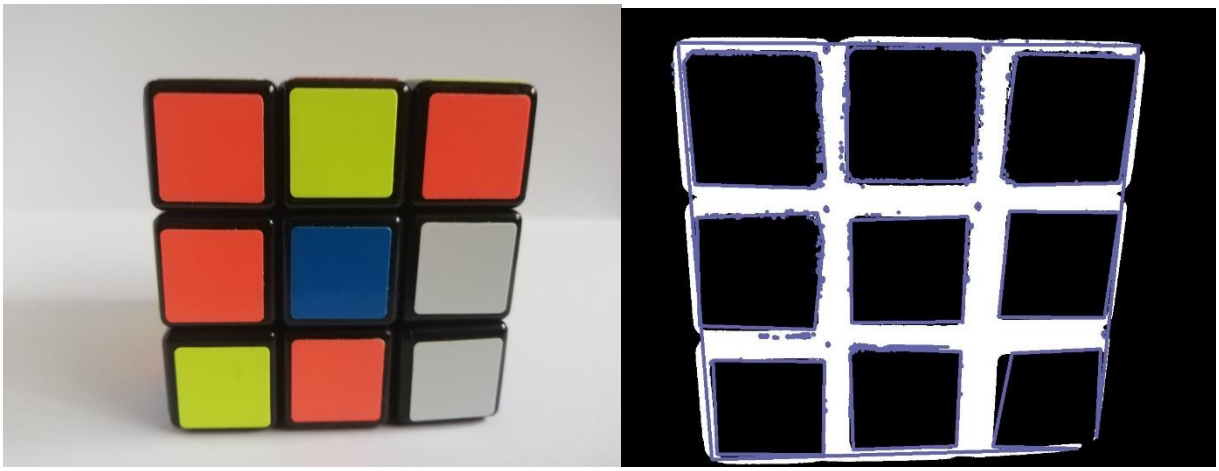
8) Ábra

Ezen a képen csak kis árnyék látható azonban ez elég nagy probléma, hiszen azt is hozzáteszi a maszkhoz. Ekkora az árnyékon még a fényerő növelés se segít. Mint látható az eredmény katasztrofális. Egyáltalán nem ismerhető fel a négyzet.



9) Ábra

De olyan képeken se biztos a siker, ami tökéletesnek látszik. Mint látható a jobb alsó sarokban nem látszik a fekete rész így a maszk se lett tökéletes. Ezért fontos, hogy látszódjon a széle is.



10) Ábra

5 Felhasználói útmutató

Ha csak alap értékekkel indítjuk el a programot akkor nincs más teendőnk csak a program mappájába kell hat darab jpg-t tenni.0-tól 5-ig legyen a nevük, szóval az első neve 0.jpg a második 1.jpg, mindegy milyen sorrendben vannak az oldalak. A képeken a kocka egyik oldala látszódjon csak, a kép legyen világos és árnyé nélküli, jól látszódjon a kocka fekete része. Elindítani a programot és megkapjuk az értékeket konzolban.

Ha kevesebb képet akarunk belerakni akkor a for M in range(6) részen a hatost át kell írni a kívánt kép mennyiségre. Ha kettőre állítjuk akkor az első 2 képet olvassa be(0 és 1 nevűt)

Ha van egy kis árnyék a képen és nem akarja megtalálni a színt segíthet ha a ”enhanced_im = enhancer.enhance(1.2)”részben az 1,2-t nagyobb érékre emeljük.

Amennyiben más színértékek van a képen akkor a ”while interv<150:” rész után az ifekben át kell írni az értékeket. Kommenttel jelöltem melyik milyen színt képvisel, RGB a sorrend.

Ha esetleg rossz színt talált egy adott kockán növelhetjük a hatékonyságot az R változó növelésével. Ez alapból 5 éréket képvisel ami 25 képpontot átlagol. De ha megemeljük 10-re akkor akár 100 képpont átlagát is figyelembe veheti. ezzel azért kell vigyázni mert ha túl nagy akkor belefog érni más színekbe is a sugara.

6 Felhasznált irodalom

[1] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html Megtekintve: (2020.05.29)

[2] https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html Megtekintve: (2020.05.30)