Dokumentáció a Rubik-kocka állapotának felismeréséhez fényképek alapján

# Bevezetés

A projekt célja a Rubik-kocka állapotának felismerése és rögzítése fényképek alapján. A script képes feldolgozni a képeket, azonosítani a színeket, majd az eredményeket egy Excel táblázatban megjeleníteni. A dokumentáció célja bemutatni a projekt megvalósításának folyamatát és az alkalmazott módszereket.

# Eszközök és technológiák ismertetése

A projekt megvalósításához az alábbi eszközöket és technológiákat használtam:

Python: A projekt fő programozási nyelve.

A Python egy magas szintű, általános célú programozási nyelv, amelyet Guido van Rossum alkotott meg, és először 1991-ben jelent meg. A Python tervezésénél nagy hangsúlyt fektettek a kód olvashatóságára és egyszerű szintaxisára, amely lehetővé teszi a programozók számára, hogy kevesebb kódsorral fejezzék ki az ötleteiket. Ennek köszönhetően a Python egy rendkívül hatékony és könnyen tanulható nyelv, amely széles körben elterjedt mind az iparban, mind az oktatásban.

OpenCV: A képfeldolgozáshoz és színazonosításhoz használt könyvtár.

Az OpenCV (Open Source Computer Vision Library) egy nyílt forráskódú számítógépes látás könyvtár, amelyet az Intel hozott létre 1999-ben, és ma széles körben használják különféle képfeldolgozási és számítógépes látási feladatok megoldására. Az OpenCV C++, Python és Java nyelveken is elérhető, de a Python interfész az egyik legnépszerűbb a könnyű használhatóságának és rugalmasságának köszönhetően.

NumPy: A numerikus számításokhoz és tömbkezeléshez használt python könyvtár.

openpyxl: Az eredmények Excel táblázatba történő írásához használt python könyvtár.

# Műszaki dokumentáció

## Könyvtárak és függvények

* OpenCV (cv2):
  + Képfeldolgozás, gamma korrekció, kontúrok keresése, színek megállapítása/összehasonlítása.
* NumPy:
  + Tömbműveletek, színek átlagolása.
* openpyxl:
  + Excel fájl létrehozása, Rubik-kocka ábrázolása, megállapított értékek megjelenítése

## Főbb függvények

adjust\_gamma(image, gamma): Gamma korrekciót végez a képen.

find\_closest\_color(color, colors): Megkeresi a legközelebbi színt az előre definiált színek közül.

rgb\_to\_hex(rgb): RGB színkódot hexadecimális formátumba konvertál.

# A script lépései

## Gamma érték beállítása:

A képfeldolgozás során nagyon fontos a megfelelő színek használata, így a scriptet kiegészítettem egy külső ciklussal. Ez a ciklus egy nagyon magas (2,5) gamma értéket állít be kezdőértékként, és a Rubik-kocka 6 oldaláról készült képeket ezzel a gamma korrekciós értékkel próbálja meg feldolgozni. A cél az, hogy minden színből 9 darabot találjon. Ha nem jár sikerrel, elkezdi a gamma értéket csökkenteni, míg nem lesz sikeres a színek felismerése, vagy el nem ér egy túl alacsony gamma értéket. Túl alacsony érték esetén egy „flag” segítségével az alapértelmezett 1.0 értékkel fut le utoljára a színfelismerés, a kimeneti fájl az így megállapított színek alapján készül el.

A képen szöveg, képernyőkép, Betűtípus, szoftver látható

Automatikusan generált leírás

1. ábra: Gamma korrekciós ciklus

## Kép beolvasása és előfeldolgozása

A beolvasott képet a script először olyan képpé alakítja, ami csak teljesen fekete vagy fehér képpontokból áll. Ez a lépés azért szükséges, mert így sokkal könnyebb megtalálni a fényképen a Rubik-kocka kontúrjait.

A képen tér, tervezés, rács, fekete-fehér látható

Automatikusan generált leírás

2. ábra: fekete-fehér Rubik-kocka.

A megtalált kontúrok segítségével egy bounding box megrajzolása következik, ami pontosan meghatározza a képen a Rubik-kocka helyzetét, majd ezeknek az értékeknek a felhasználásával az eredeti képből kivágja a hasznos területet, így a kép többi részével nem kell a későbbiekben foglalkozni.

A képen Mechanikus kirakójáték, játék, Rubik-kocka, puzzle látható

Automatikusan generált leírás

3. ábra: Kontúrok és bounding Box.

## A kivágott Rubik-kocka képének feldolgozása:

Miután elkészült a kivágott kép, a script alkalmazza erre a képre a gamma korrekciós függvényt.

A gammakorrekció után a képet 2 vízszintes és 2 függőleges vonal segítségével a script 9 egyenlő részre osztja, majd ezeket a kép részleteket egyesével, külön képekként dolgozza fel.

A képen Színesség, színek, paletta, művészet látható

Automatikusan generált leírás

4. ábra: 9 egyenlő területre osztás.

A képen Színesség, Téglalap, színek, művészet látható

Automatikusan generált leírás

5. ábra: a 9 különálló cella.

A Rubik-kocka fehér oldalán középen található a termék logója. Ez a színekkel teli logó vezetett ahhoz a megoldáshoz, hogy mind a 9 képből egy kisebb részletet vág ki a script a bal felső részből a cella színének megállapításához. (És ahhoz is, hogy kötött a kockáról készült képek elkészítésének módja)

## Színek felismerése:

Miután megvan a kilenc színminta, a mintán található színek átlagainak megállapítása történik, RGB színkódokkal.

A kiszámolt értéket a script összehasonlítja egy színtáblázattal, ezután a táblázatban meghatározott színkódok közül megkeresi, melyikhez áll a legközelebb. A megtalált színt eltárolja egy numpy tömbben.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

6. ábra: színkódok

A színkódok meghatározásához a Rubik-kockáról készült első felvételsorozatot használtam. A színek 9 példányának RGB értékeit egy Excel táblázatban rögzítettem, majd az értékek átlagait használtam fel a „colors” értékeinek megadásához. (repo\ színértékek.xlsx)

A képen szöveg, képernyőkép, szám, sor látható

Automatikusan generált leírás

7. ábra: Excel tábla a színkódok megállapításához.

Miután mind a hat kép feldolgozása megtörtént, minden színből kilenc darabnak kell lennie, valamint a hat középső cellának mind egyedi színnel kell rendelkeznie, itt nem lehet színismétlődés.

## Log fájl készítése:

Kiegészítettem a scriptet egy szöveges dokumentum elkészítésével, amiben megtalálható a felismert színek adatstruktúrája két verzióban. Az első a színek RGB kódjait tartalmazza, míg a második a színek neveit.

A log fájl továbbá tartalmazza a megtalált színek darabszámát, a középső cellák színeit és egyediségét, valamint a színfelismerés során használt gamma értéket.

## Eredmény prezentálása Excel táblázat használatával:

A script utolsó fázisában készít egy Excel fájlt, ahol a teljes Rubik-kocka ábrázolása történik 2 dimenzióban.

Az eredmény ellenőrzésének, hibák keresésének megkönnyítése érdekében három módon jeleníti meg a megállapított értékeket.

Az első részben a cellákat az RGB színkódnak megfelelő színnel tölti fel a script,  
a második rész a színek neveit tartalmazza,  
a harmadik rész pedig az RGB színkódot.

A „kiterített kocka” alatt a színösszesítés és a gamma érték is megtalálható.

Az Excel fájl elkészítése után a script a futtatásának végéhez ér.

# A script használata:

## A Rubik-kocka fényképezése:

A script használatához szükség van a Rubik-kocka hat oldaláról készült fényképekre.  
A fényképeknek egységes színű háttérrel kell rendelkeznie, a nagyon erős árnyékok, színátmenetek, mintázatok hatására a kontúrok megtalálása nehézkessé válhat.

A „kiterített kocka” ábrázolásmódnak megfelelően kell elkészíteni a képeket a Rubik-kockáról, viszont az első képnek a logóval ellátott fehér oldalnak kell lennie.

A zöld, narancssárga, piros és kék oldalak csak a teljesen kirakott esetben vannak a 9. ábrán látható pozíciókban, a kocka más állapotában más pozícióban is lehetnek.

Fehér oldal helyes tájolása:

A képen szöveg, képernyőkép, tér, sor látható

Automatikusan generált leírásA képen rajzfilm látható

Automatikusan generált leírás

9. ábra: 2D-s ábrázolás, "kiterített kocka"

8. ábra: Rubik's logó tájolása.

A fényképek elkészítésének folyamata:

1. kép: a fehér oldal, a Rubik’s logó a 8. ábrán látható tájolásban van.

2. kép: a fehér oldal fölötti rész, a kockán egyet kell fordítani magunk felé.

3. kép: a fehér oldal a kiinduló pont, a balra eső oldalról kell képet készíteni, ehhez a kockán jobbra kell fordítani egyet.

4. kép: a fehér oldal ismét a kiinduló pont, a jobbra eső oldalról kell képet készíteni, ehhez a kockán balra kell fordítani egyet.

5. kép: a fehér oldal alatti oldal, a kockán a fehér oldaltól számítva egyet kell fordítani fölfelé.

6. kép: a sárga oldal, a kocka alja (ha a fehér oldalt tekintjük a tetejének). A fehér oldaltól számítva kettőt kell fordítani a kockán fölfelé.

## Képfájlok elhelyezése:

A képek számára létre kell hozni egy mappát, majd a mappában a kép sorszámának megfelelően kell a képet elnevezni. (pl: 1.jpg, 2.jpg, 3.jpg, 4.jpg, 5.jpg, 6.jpg)

A scriptben meg kell adni a mappa nevét a 69. sorban található mappa változóban.



## Script futtatása:

A script futtatásához szükség van a python telepítésére, valamint pip segítségével telepíteni kell az opencv-python könyvtárat (ez telepíti a numpy-t is) és az openpyxl könyvtárat.

Ehhez Command prompt-ban az alábbi sorokat kell beírni:

pip install openpyxl

pip install opencv-python

Eltérő telepítés esetén a futtatás eltérhet, Windows-os környezetben a Command prompt-ból a futtatás történhet az alábbi módon:

python rubik9.py

A script futtatásának végén a Command prompt ablakban a „kész” felirat jelenik meg.

A log fájl és az eredményt tartalmazó Excel fájl a képeket tartalmazó mappába kerül.

# A színfelismerés kiértékelése:

ADATOK KIÉRTÉKELÉSE

további fejlesztési lehetőségek

* GUI, argumentumok, képek tájolásának felismerése, az adott állapotból a kirakáshoz szükséges lépések megadása, „debug mód” ( minden lépésnél megmutatja, hogy mi történt a képpel)

fejlesztési folyamat során tapasztalt dolgok