Beadandó

Téma:

Objektum színének meghatározása

Készítette:

Nagy Péter

L242N0

Tartalomjegyzék:

[Bevezetés 2](#_Toc61468963)

[Felhasználói dokumentáció 2](#_Toc61468964)

[Telepítés 2](#_Toc61468965)

[Program indítása 2](#_Toc61468966)

[A program működése 2](#_Toc61468967)

[Fejlesztő dokumentáció 3](#_Toc61468968)

[A program indítása 3](#_Toc61468969)

[A program működése 3](#_Toc61468970)

[Kontúrok keresése 4](#_Toc61468971)

[Kontúrok vizsgálata 5](#_Toc61468972)

[Tesztelés 5](#_Toc61468973)

[Összegzés 7](#_Toc61468974)

# Bevezetés

A választott téma teljes címe:

Előre definiált lehetséges színek felhasználásával egy homogén (pl. fehér) háttér előtti egyszínű, de nem feltétlen homogén objektum színének meghatározása nem ideális körülmények között (pl. rossz fény- viszonyok, zavaró árnyék stb.).

# Felhasználói dokumentáció

## Telepítés

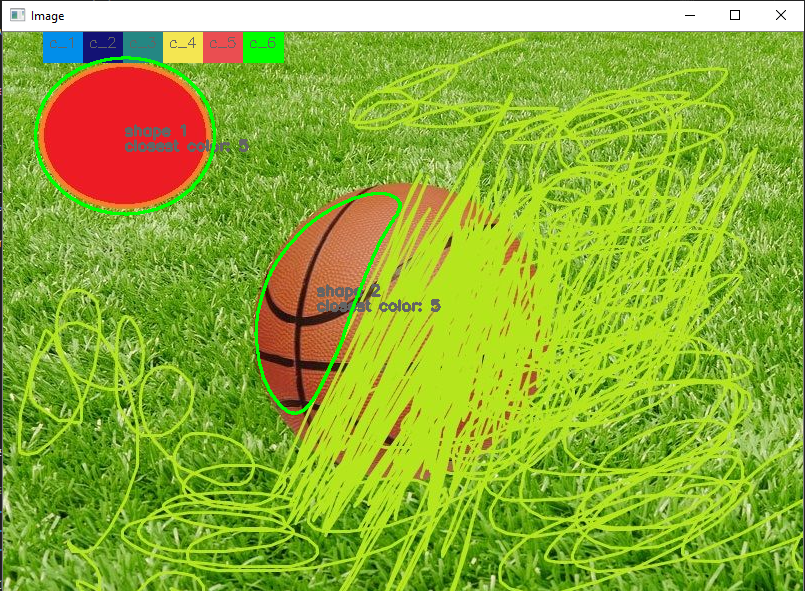
1. A program githubról tölthető le, melynek címe: <https://github.com/nagypeterpal/gepilatas>   
   Másoljuk be egy tetszőleges könyvtárba.
2. A program futtatásához a Python programozási nyelvet telepíteni kell a számítógépre.  
   Letölthető innen: <https://www.python.org/downloads/>  
     
   Én a 3.8.5-ös verzió hasznátam, tehát azzal működik biztosan.   
   Más verziókkal való kompatibilitást nem vizsgáltam.
3. Egyéb csomagok telepítése: pl. pip install ‘packagename’ paranccsal.  
     
   Szükséges package-ok:  
   argparse, imutils, cv2, os , numpy

## Program indítása

Amennyiben a python telepítése sikeres volt, csak menjünk bele abba könyvtárba melybe a letöltöttük a githubról a fájlokat és kattintsunk a “start.bat” fájlra és a program indul.

## A program működése

A program alapvetően betölti egyesével az “images” könyvtárból a képeket (átméretezést nem végez, más képek használatat esetén célszerű őket max. 800\*600 vagy más a monitoron is jól látható méretre kicsinyiteni) majd feltűnő objektumokat keres rajtuk és megkeresi hogy ezek színei melyik előre definiált színhez állnak a legközelebb. Ezek a színek a colors.txt fájlban vannak definiálva soronként BGR formátumban. Az egyes képek kirajzolása után gombnyomás után halad tovább a program a következő képre.



A képen látható, hogy a szoftver megtalálja az elütő, markáns területeket és meghatározza a hozzájuk legközelebb eső színt a fenti skálából.

# Fejlesztő dokumentáció

## A program indítása

A program a start.bat batchfile-al indul, ami paraméterezetten hívja meg a programot a python környezetben: python center\_of\_shape.py --image images --colors colors.txt

Paraméterként a könyvtár és a színeket tartalmazó fájl adható meg.

## A program működése

A program betölti az előre definiált színeket majd egyesével a képeket a paraméterként kapott könyvtárból és elvégzi a feldolgozásukat a következő lépések szerint.

Lépés 1.: Klónozás

A betöltött képből készítünk egy klónt, amin majd a műveleteket fogjuk elvégezni. Az eredeti képet azért nem módítjuk, mert a vizsgálat végén ezt rajzolujuk ki, valamint:

* megjelenítjük rajta az előre definiált és betöltött színeket
* kirajzoljuk rá a kontúrokat
* a kontúrok középpontjánál kiírjuk az azonoítojuk és a színkódot amihez a színük a legközelebb van

Lépés 2.: Eredeti képre a betöltött színek kirajzolása

Ciklussal végigmegyünk a betöltött tömbön és kirajzoljuk a négyzeteket egymás után és ellátjuk őket a c\_ kezdetű futó azonosítóval.

### Kontúrok keresése

Lépés 3.: Homályosítjuk a képet

A cv könvytár GaussianBlur() funkcióját használjuk, amely egy lokális értékkészlet transzormáció. Ezzel a szürővel a közelebbi szomszédok nagyobb súllyal szerepelnek. A következő paraméterekkel hívjuk meg:

* Image\_clone: ezen végezzük a transzformációt
* (105, 105): a kernel mérete: ezt jó nagyra állítottam, mert a legnagyobb kiterjedésű elötő objektumot keressük (természetesen pozitívnak és a páratlannak kell lennie)
* 0: standard deviáció

Lépés 4.: Kvantálás

Az elhomályosított képet megpróbáljuk két színklaszterre felbontani, tehát a két leggyakoribb szinközépértéket meghatározni és az adott kép képpontértékeit erre a két színre redukálni. Ehhez először a kép kétdimenziós tömbjét egydimenziós tömbé alakítjuk, majd a képpont értékeket pedig lebegőpontos számmá, mert a k-mean függvény ezzel az adattípussal dolgozik.

Meghatározzuk a kritériát, 2-ős kritéria:

**cv.TERM\_CRITERIA\_EPS + cv.TERM\_CRITERIA\_MAX\_ITER** - iteráció abbahagyása ha elérjük a kellő pontosságot vagy a max iterációt

Végrehajtjuk a kmeans() függvényt a következő paraméterekkel:

* Flattened\_img: amin végrehajtjuk a függvényt
* cluster\_number=2: hány klaszterre bontsuk szét az egydimenziós tömböt
* Kritéria: lásd előbb.
* 10: a kisérletek száma ahányszor a függvény próbálja a klaszterezést, a végső eredmény a legkompaktabb lesz
* KMEANS\_RANDOM\_CENTERS: véletlenszerű középponttal kezd, majd közelíti

A kapott eredménytömböt végül visszaalakítjuk 2 dimenziós tömbbé, melynek minden eleme valamelyik klaszter érték.

Lépés 5.: Szürkeárnyalatos konverzió

Az előzőleg kapott klaszterezett képet szürkeárnyalatossá alakítjuk a cvtColor függvénnyel. Ez azt jelenti hogy a 3 színcsatorna helyett – a súlyozott átlag számításával – egy színcsatornára térünk át. A függvényt a következő paraméterekkel hívjuk meg:

* Reshaped: az előző transzformáció eredménye. Ezen hajtjuk végre a konverziót.
* cv2.COLOR\_BGR2GRAY: BGR színtérből alakít szürkeárnyalatos

Lépés 6.: Bináris képpé alakítás

A szürkeárnyalatos képet egy bináris képpé alakítjuk egy küszöbölő függvénnyel. A cv2.threshold függvényt a következő paraméterekkel hívjuk meg:

* Gray: a szürkeárnylatos kép a bemenet
* 0: küszöb, ez nincs használatban, mert adaptív módban használjuk a függvényben
* 255: az érték ,mely a bináris küszöbölés másik értéke lesz
* cv2.THRESH\_BINARY+cv2.THRESH\_OTSU: itt határozzuk meg hogy bináris lesz a küszöbölés és adaptív: az OTSU algoritmust használja jelen esetben.

Lépés 7.: Kontúrkeresés

Megkeressük a kontúrokat a legegyszerűbb módon listába és terület szerint csökkenően rendezzük. Csak az első alakzattal foglalkozunk. A lépés a cv2.findContours függvényt használja ( ami Satoshi Suzuki által írt: digitális bináris képek határkövetése topológiai strukturális analízissel c. munkája alapján készült). A paraméterek a következők:

* Thresh: bemeneti kép
* cv2.RETR\_LIST: listaként kapjuk vissza kontúrokat
* cv2.CHAIN\_APPROX\_SIMPLE: csak sarokpontos tárolás

### Kontúrok vizsgálata

Ezen a ponton megvannak a kontúrjaink, s ezekkel dolgozunk tovább.

Lépés 8.: Kontúrok középpontjának meghatározása

A középpontok meghatározásához a CV által biztosított ‘moments’ objektumokat használhatjuk, melyek valamiféle kép intenzitás alapú súlyozott átlagok, melyek segítségével a kontúrok különböző tulajdonságaihoz férhetünk hozzá. Így juthatunk el a középpontajikhoz is.

Lépés 9.: Legközelebbi szín keresése

Mindegy egyes Kontúrnál mintát veszünk a homályosított képből (Lépés 3.) a 8. Lépésben számolt ponton. Ezután egyesével végigmegyünk az előre betöltött színeken és meghatározzuk csatornánként a különbséget:

* + - 1. csatornánként kivonjuk a próbaszínt az aktuális képpontból és vesszük az abszolútértékét:  
         tol\_g=abs(int(pr\_g)-int(g))  
         tol\_b=abs(int(pr\_b)-int(b))  
         tol\_r=abs(int(pr\_r)-int(r))
      2. összeadjuk a 3 abszolútértéket:  
         tol\_arr.append(tol\_g + tol\_b + tol\_r)

majd ezek abszolút értékét összeadjuk és egy tömbben letároljuk. Végül meghatározzuk a tömb legkisebb elemének sorszámát.

Lépés 10.: Feliratozás

Az eredeti képre felrajzoljuk a kontúrokat, azok neveit ‘shape ’ + sorszám. Alatta lévő sorba pedig a hozzá legközelebb eső szín sorszámát.

# Tesztelés

A tesztelés a következőképpen zajlott: egyesével kirajzoltuk a képeket és dokumentáltuk a gép általi színmeghatározást a külső szemlélő személy általi meghatározással összevetve:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Kép sorszáma | Gép  mit talált | Gép  szín | Személy  mit talált | Szem.  szín | Különbség | Megjegyzés |
| img001 | Jobb 3 fekete alakzat egyben | 2 | Bal 2 egybefüggő alakzat | 2 | Más formát talál meg | Homályosítás az oka |
| img002 | Hektor | 2 | Hektor | 2 | - |  |
| img003 | Jobb oldali terület | 3 | Hektor | 2 | Más formát talál meg | Homályosítás az oka |
| img004 | Hold | 4 | Hold | 4 | - |  |
| img005 | Piros kör | 5 | Piros kör | 5 | - |  |
| img006 | Kép alsó fele | 3 | Gomba, szín: 5 | 5 | Más formát talál meg |  |
| img007 | Piros téglalap | 5 | Piros téglalap | 5 | - |  |
| img008 | Narancs | 4 | Narancs | 4 |  |  |
| img009 | Jobboldali barack | 5 | Jobboldali barack | 4 |  | Szín kb. kettő között féluton |
| img010 | Holló | 2 | Holló | 2 |  |  |
| img011 | Hangya | 5 | Hangya | 5 |  |  |
| img012 | Kép alsó része | 6 | Piramis | 6 | Más formát talál meg | Homályosítás és közeli színek |
| img013 | Föld | 3 | Kék hordó | 1 |  | A hordó kicsi a kép méretéhez képest |
| img014 | Legó | 3 | Legó | 3 |  |  |
| img015 | Legó | 3 | Legó | 3 |  |  |
| img016 | Kép alsó része | 5 | Uluru szikla | 5 | Más formát talál meg | Homályosítás és közeli színek |
| img017 | Fotós | 2 | fotós | 2 |  |  |
| img018 | Kép alsó része | 5 | Legnagyobb ballon | 5 | Más formát talál meg |  |
| img019 | Avokádó + kis keret | 5 | Avokádó | 4 | Más formát talál meg | Homályosítás miatt |
| img020 | Egér | 3 | Egér | 3 |  |  |
| img021 | Ház | 3 | Ház | 3 |  |  |
| img022 | Jobb oldali madár | 2 | Jobb oldali madár | 3 | Más szín | színek közel egymáshoz |
| img023 | Ember | 5 | Ember | 5 |  |  |
| img024 | Kép felső része | 5 | Nap | 3 | Más forma | ilyen kis objektumot nem tud megtalálni |
| img025 | Kép alsó része | 3 | Bálna | 3 | Más forma | közeli színek |
| img026 | egész kép | 3 | Párduc | 4 | Más forma | Nagyon egybeolvad |
| img027 | zsiráf | 4 | zsiráf | 4 |  |  |
| img028 | bal oldali cseresznye | 5 | bal oldali cseresznye | 5 |  |  |
| img029 | 3 cseresznye egybe | 3 | jobb oldali cserezsnye | 5 | Más forma | összemosódás |
| img030 | 3 madár egyben | 5 | 3 madár egyben | 5 |  |  |
| img031 | repülő + hegyek | 3 | repülő | 3 | más forma | összemosódás |
| img032 | ég lilás része | 2 | fa | 2 | más forma | kicsi objektum |
| img033 | madár | 5 | madár | 5 |  |  |
| img034 | ég a fák között | 4 | ég a fák között | 3 | más szín | közeli színek |
| img035 | birka, hátsó birka | 4 | birka, hátsó birka | 4 |  |  |
| img036 | kalaptoll+ háttér egy része | 2 | kalaptoll | 2 | más forma | összemosódás |
| img037 | kalaptoll+ háttér egy része | 2 | kalaptoll | 2 | más forma | összemosódás |
| img038 | férfi+bútor | 2 | férfi+bútor | 2 |  |  |
| img039 | fa + rét | 3 | fa | 5 | más forma | itt 3 részre kellene klaszterezni |
| img040 | majom orrának jobb fele | 1 | majom orrának piros része | 5 | más forma |  |
| img041 | majom orra | 3 | majom orra | 3 |  |  |
| img042 | teniszlabda + háttér | 5 | teniszlabda | 6 | más forma | kicsi az objektum |
| img043 | egész kép | 3 | lila gomb | 2 | más forma | kicsi objektum |
| img044 | zöld paprikák egyben | 3 | zöld paprikák egyben | 3 |  |  |
| img045 | bal oldali paprika | 3 | sötétebb paprika | 3 | más forma | kisebbnek tűnik |
| img046 | ember sötét ruhája | 2 | ember sötét ruhája | 3 |  |  |
| img047 | piros predátor | 5 | piros predátor | 5 |  |  |
| img048 | kép bal része | 3 | fehér híd | 4 | más forma |  |
| img049 | lány egészben | 3 | lány egészben | 3 |  |  |
| img050 | hölgy mögött terület | 3 | hölgy | 3 | más forma |  |
| Összesítés: |  |  |  |  |  |  |

# Összegzés

Úgy látom alapvetően a formadetektálásban van eltérés, a színek meghatározás nagyjából stimmel. Egy adott képhez még viszonylag könnyű lenne úgy alakítani a paramétereket, hogy hasonló eredményeket hozzon, de ennyire eltérő képeknél nem tudom létezhet-e ezzel a megoldási módszerrel egy végső paraméterbeállítás, valószínűleg valami tanuló algoritmus irányába kellene továbbfejleszteni a szoftvert..