Gesztusvezérelt egér szín- és alakzatdetektálással

Bézi Patrik

2021. 09. 16.

Tartalomjegyzék

1. Bevezetés

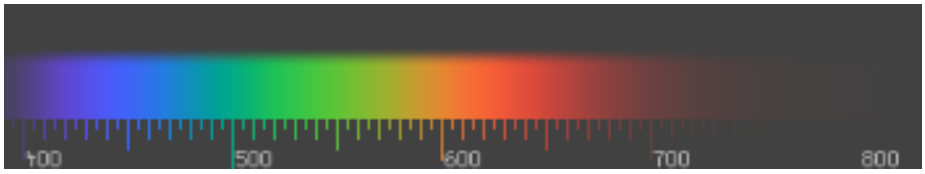
A számítógépes programozás és a képfeldolgozás olyan szintre fejlődött az elmúlt évek során, melynek során könnyen fejleszthetőek olyan szkriptek, amelyek a gépek mindennapi felhasználását elősegítik.
A dolgozatom során egy olyan szoftvert fejlesztek, amely a bemeneti képet szín és egyszerű alakzatszűrő segítségével képes feldolgozni úgy, hogy azzal az egér mozgását lehessen befolyásolni. Az ötlet egy egyszerű elven alapszik, miszerint legyen bármilyen tárgy alkalmas arra, hogy a gép egere felett könnyedén át lehessen venni az irányítást, legyen szó prezentálásról, szórakozásról, vagy egyszerű kényelmi szempontokról.
A szoftver legfőbb követelménye a valós idejű képfeldolgozás, vagyis hogy az adott tárgy és a számítógép egerének mozgása a lehető legnagyobb szinkronban történjenek. Ennek a sebességnek az eléréséhez fontos kiválasztani a megfelelő felbontást és a megfelelő műveleteket, amelyek nem okoznak különösebb gondot a számítási igényükkel még egy gyengébb teljesítményű gépnek sem.
Legelőször ki kell választani azt az eszközt, amellyel az irányítást el szeretnénk végezni. Ha ez megvan, el kell érni, hogy a kamera felismerje azt és lekövesse a mozgását.
A felismerés pontosságának fejlesztéséhez maszkolási eljárásokat is alkalmazni fogok, amelyek működését össze fogom hasonlítani a dolgozat készítése során.
A szoftver fejlesztése **Python 3** nyelven történt **Ubuntu 20.04** környezetben. A megoldáshoz kizárólag az **OpenCV** keretrendszer 4.5.2-es verzióját, illetve a **numpy** csomagot használtam fel.

2. Színmodellek

A háttértudás leírásához az alábbi irodalmakat használtam fel: [1] [2] [3]

2.1. Színek

Az elektromágneses hullámok közül nem mindegyiket érzékeljük szemünkkel, pedig természetük azonos. Az átlagos emberi szem kb. a $3,810^{-7}m$ és $7,610^{-7}m$, azaz a 380 nm és 760 nm közötti hullámhossztartomány érzékelésére képes: az ebbe a tartományba eső elektromágneses sugárzást nevezzük fénynek. Ez a látható tartomány természetesen emberről emberre változik. A különböző hullámhosszaknak a tiszta színek felelnek meg, amelyet az 1. ábra szemléltet.

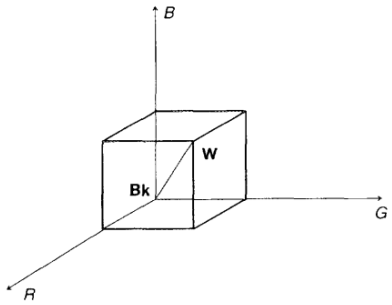


1. ábra. A látható spektrum és a fény hullámhossza nanométerben.[1]

A színek jelentősége hatalmas az egész informatikában. A belőlük elkészített színmodellek különböző feladatok elvégzésére tesz alkalmassá. Egy rajzolóprogram például nem ugyanazt a modellt használja, mint a nyomtatók. A következő két alfejetben bemutatom az RGB-, illetve a feladat megoldására használt HSV modelleket.

2.2. RGB modell

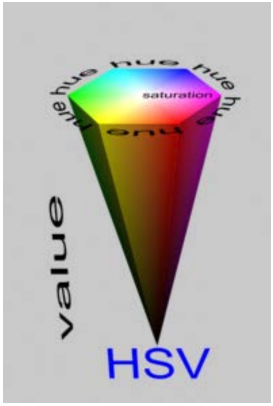
Színmodellnek nevezünk egy háromdimenziós testet, amely a 3 koordináta partikuláris leírásának segítségével leír egy színt.
Az RGB-re épülő talán a legismertebb ilyen modell, amely a vörös, zöld és kék színek keverésével segít meghatározni a színeket. A skála [0-255] intervallumon belül terjed mindhárom szín esetében. Minél nagyobbak ezek az értékek, az eredmény annál világosabb és minél jobban közelít a 0-hoz, annál sötétebb lesz.



2. ábra. Az RGB színkocka.[3]

2.3. A HSV modell

A HSV a hue(árnyalat), saturation(telítettség) és a value(fényerő) szavak rövidítéseiből tevődik össze. Ezek kombinációja határozza meg a színeket egy adott képen.
Miért ezt a kódolást használom? Mert egy adott színt sokkal könnyebb leírni ezen kódolás segítségével. Az elsődleges elképzelés a feladat megoldásáról, hogy minden értéknek legyen egy alsó és egy felső határa, amelyeket közötti színeket figyelembe fogok venni. Ha például bizonyos fényviszonyok között én egy sárga toll mozgását szeretném lekövetni, akkor ahogy mozgatom, a fent említett értékek folyamatosan változhatnak. Ebből kifolyólag célszerű megadni olyan határértékeket, amelyek között a sárga a még folyamatosan változó körülmények esetén is felismerhető lesz.



3. ábra. A HSV modell.[3]

2.3.1. Fogalmak

- Árnyalat: ez adja meg tulajdonképpen a fény hullámhosszát a spektrumban. Az OpenCV-ben ez az érték [0-179] tartományban terjed
- Telítettség: A szín tisztaságát adja meg. Ez annyit jelent, hogy az adott színárnyalat telítettebb formában élénkebb, míg semlegesebb formában egyre jobban közelít a szürkeárnyalathoz. Ez az érték OpenCV-ben [0-255] tartományban terjed.
- Fényerő: Egy szín fényerejét adja meg. Ez az érték OpenCV-ben [0-255] tartományban terjed.

A fent említett három tulajdonság segítségével kiszűrhetővé válik az a spektrum, amelyet figyelembe szeretnénk venni a szoftver futása alatt. A legelső lépés ehhez, hogy az OpenCV beépített függvényét felhasználva az alap BGR kódolású képet át kell konvertálni HSV kódolásra. Ennek matematikai levezetése az OpenCV dokumentációja alapján a következőképpen alakul:

$$V \leftarrow max(R, G, B)$$

$$S \leftarrow \begin{cases} \frac{V - max(R,G,B)}{V} & V \neq 0 \\ 0 \end{cases}$$

$$H \leftarrow \begin{cases} 60(G-B)/(V - min(R,G,B)) & V = R \\ 120 + 60(B-R)/(V - min(R,G,B)) & V = G \\ 240 + 60(R-G)/(V - min(R,G,B)) & V = B \\ 0 & R = G = B \end{cases}$$

A HSV-s kép segítségével készíthető egy maszk, amelyet felhasználva figyelmen kívül lehet hagyni a szükségtelen színkódú pixeleket.

3. Maszkolás

4. Kontúrok

A kontúrok fontos szerepet töltenek be a projektem során. Egyrész segítenek meghatározni egy adott tárgy alkazatát, másfelől a kontúr által befoglalt terület nagyságát befolyásolva megadhatóak olyan paraméterek, amelyek segítenek a háttérben lévő zajok szűrésében. Kontúrnak nevezzük azokat a helyeket, ahol az intenzitás értéke nagy mértékben hirtelen megváltozik. Ez a dolgozatomban segít meghatározni az objektum határvonalát. Éldetektáláskor nincs nagy jelentősége a színeknek, úgyhogy a folyamat előtt érdemes szürkeárnyalatossá tenni a képet.

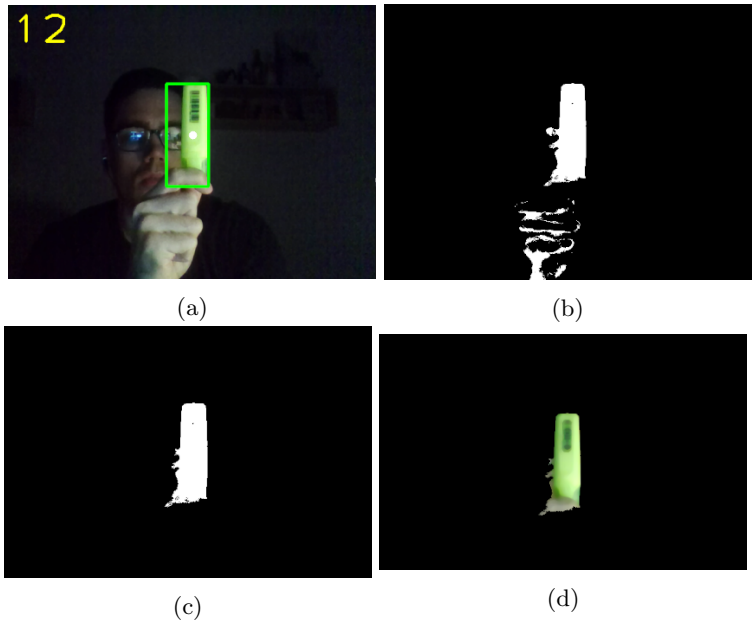
5. Alakzatok, befoglaló terület nagysága

A fals pozitív értékek csökkentésének az egyik módjaként megkíséreltem figyelembevenni az vezérlőnek használt tárgy alakját. Hogy ezt kiderítsem, egyszerűen csak megszámolom a kontúr által bezárt szögek számát. A kontúrok meghatározásával lehetőség nyílik megfigyelni a detektált objektumok alakzatát egy egyszerű trükkel. Ez nem más, minthogy az egybefüggő testek csúcspontjait megszámlálom, ezáltal meghatározhatóvá válik, hogy milyen alakzatot detektál a szoftver.

6. Marás, foltok egyesítése, blurring effekt [3]

7. Követés középponti elmozdulás alapján

Ha megtaláltam az adott objektumot, amivel a feladatot végre szeretném hajtani, akkor annak középpontját veszem és a továbbiakban ennek a pontnak az elmozdulása fogja meghatározni az egér mozgását.



(a) (b)

(c) (d)

4. ábra. Itt látható a szoftver működése. Az (a) az objektum középpontját, a (b) a HSV szűrő segítségével létrehozott maszkot, a (c) a kontúr által befoglalt területi szűrőt, a (d) pedig a szoftver által látható képet ábrázolja.

8. Egér vezérlése

9. Kiértékelés, Összegzés, Korlátok, Lehetőségek

10. A program telepítése

10.1. Futtatáshoz szükséges szoftveres követelmények

Amire szükség van a program futtatásához:

- Python 3.8.10
- OpenCV 4.5.2 (én a contrib verziót használtam a projekt elkészítése során)
- numpy csomag
- egy kamera, amiről a számítógép tudja olvasni a képi információkat

A szoftvert Ubuntu 20.04-es környezetben fejlesztettem, amelyen már alapból megtalálható a Python megfelelő verziója. A hozzá szükséges csomagokat a pip csomagkezelő segítségével lehet letölteni az alábbi parancsokkal:

```
python3 -m pip install numpy
python3 -m pip install opencv-contrib-python
```

Ezek után a program futtathatóvá válik.

10.2. A program futtatása

A szoftvert alapvetően kétféle módban lehet futtatni. Az első a beállítási mód, amely lehetővé teszi a felhasználó számára, hogy a HSV kódolást saját magának beállítsa, lehetővé téve számára, hogy saját eszközt is kiválaszthasson a vezérlésre. A másik mód az, amelyikben a rendszer már a megfelelő HSV-s adatokat beolvasva és felhasználva élesben működik.

Irodalomjegyzék

[1] Berta Miklós, Farzan Ruszlán, Giczi Ferenc, Horváth András - Fizika mérnököknek, 298-300 oldal ,2006

[2] Richard Szeliski – Computer Vision: Algorithms and Applications, p90, 2010

[3] Levkowitz, H. Herman, G.T. - CVGIP: Graphical Models and Image Processing Volume 55, Issue 4, July 1993, Pages 271-285

[4] OpenCV Documentation - https://docs.opencv.org/4.5.3/de/d25/imgproc_color_conversions.html

[5] Werner Purgathofer - Einführung in Visual Computing – Kapitel 4, TU Wien, 2017