Gesztusvezérelt egér szín- és alakzatdetektálással

Bézi Patrik

2021.11.02.

Tartalomjegyzék

1. Bevezetés

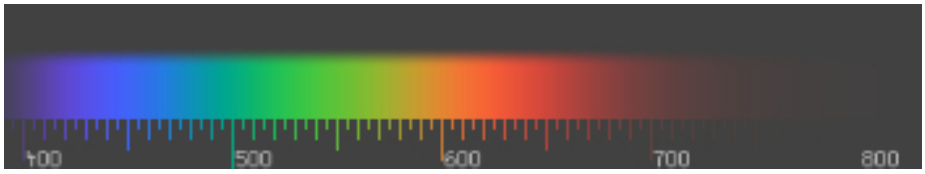
A számítógépes programozás és a képfeldolgozás olyan szintre fejlődött az elmúlt évek során, melynek következtében könnyen fejleszthetőek olyan szkriptek, amelyek a gépek mindennapi felhasználását könnyebbé tehetik.
A dolgozatomban egy olyan szoftvert fejlesztek, amely a bemeneti képet szín és egyszerű 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 alkalmaztam, amelyek működését le is teszteltem a dolgozat készítése közben.
A szoftver fejlesztése **Python 3** nyelven történt **Ubuntu 20.04** környezetben. A megoldáshoz az **OpenCV** keretrendszer 4.5.2-es verzióját, illetve a **numpy** csomagot használtam fel. Ezek mellett szükség volt az **autopy**, illetve a **screeninfo** csomagokra, amelyek közül az egyikkel az egér mozgatását oldottam meg a tárgyam képi koordinátáinak lekövetésével, míg a másik a monitor felbontását figyelembe véve rendszerezve helyezi el a képernyőn a felugró ablakokat.
A szükséges csomagok telepítésének módját a **"A program telepítése és működtetése"** című fejezetben írtam le.

2. Színmodellek

A háttértudás leírásához az alábbi irodalmakat használtam fel: [1] [2] [3] [4] [5]

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.

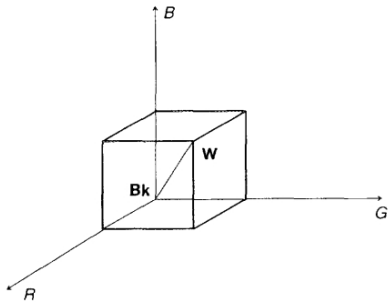


1. ábra. A látható spektrum és a fény hullámhossza nanométerben.[1]

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 alfejezetben bemutatom az RGB-, illetve a feladat megoldására használt HSV modelleket.

2.2. RGB modell

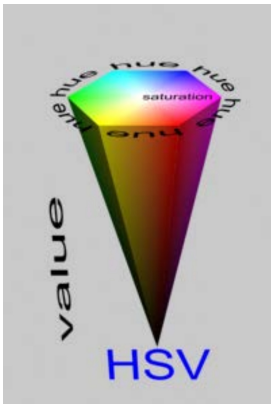
Színmodellnek nevezünk egy háromdimenziós testet, amely a 3 koordináta partikuláris leírásának segítségével leír egy színt.
Az RGB-re épülő talán a legismertebb ilyen modell, amely a vörös, zöld és kék színek keverésével segít meghatározni a színeket. A skála [0-255] intervallumon belül terjed mindhárom 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.
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, amelyek 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.[5]

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 lesz, 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:

$$V \leftarrow max(R, G, B) \tag{1}$$

$$S \leftarrow \begin{cases} \frac{V - max(R,G,B)}{V} & V \neq 0 \\ 0 \end{cases} \tag{2}$$

$$H \leftarrow \begin{cases} 60(G-B)/(V - min(R,G,B)) & V = R \\ 120 + 60(B-R)/(V - min(R,G,B)) & V = G \\ 240 + 60(R-G)/(V - min(R,G,B)) & V = B \\ 0 & R = G = B \end{cases} \tag{3}$$

A HSV-s kép segítségével készíthető egy maszk, amely kezdetben egy a bemenet méretével megegyező nagyságú kép. Ennek küszöbértékeit határozza meg a 3-3 (minimális és maximális) *H,S,V* érték. Ezt felhasználva figyelmen kívül lehet hagyni a szükségtelen színkódú pixeleket.

3. Maszkolási eljárások

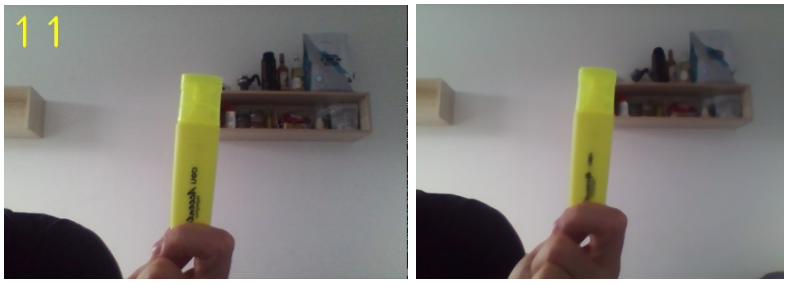
A következő fejezetben bemutatom a maszkolási eljárásokat, amelyeket alkalmaztam a szoftveremben. Ennek elkészítéséhez az alábbi forrásokat használtam fel:[4] [5] [6] [7]

3.1. Előfeldolgozás

A könnyebb szűrés és az esetlegesen felmerülő fals pozitív eredmények kiküszöbölése érdekében a bemeneti képet egy homályosító eljárásnak vetettem alá. Ehhez mediánszűrőt alkalmaztam, mely a következő módon működik:
Legyen *I(x,y)* a bemeneti kép, *J(x,y)* a rá vonatkozó értékkészlet transzformáció, *S(x,y)* pedig a vizsgált képrészlet. Ezek alapján legyen a vizsgált méretű képrészlet alapján képzett számsorozat mediánja az új intenzitás érték:

$$J(x,y) = medI(i,j)|I(i,j) \in S(x,y) \tag{4}$$

A medián szűrővel az a cél, hogy az összefüggő színeket egybemosva csökkenjen a háttérzaj mértéke a HSV határértékek beállítása alatt is már.



(a) (b)

4. ábra. A kép átalakítása a medián szűrő alkalmazásával.

3.2. 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 folyamathoz nyugodtan lehet használni a már korábban elkészített maszkot, amelynek küszöbértékei egyelőre csak a megfelelő HSV értékekhatárok alapján kerülnek meghatározásra.

Az élek felismerésére az Opencv *findContours* függvényét használtam fel a dolgozat során.

5. ábra. Kontúrdetektálás az OpenCV *findContours* függvényével.

3.3. A befoglalt terület

A szoftver végigmegy valamennyi a maszkon fellelhető "folton" és megkeresi azok kontúrjait. Ezzel meghatározhatóvá válik az általuk befoglalt terület nagysága. Ennek jelentősége, hogy a képen egy bizonyos területnél kevesebbet, illetve többet foglaló testek ne jelenjenek meg zajként. A későbbi, középponti elmozdulással kapcsolatos fejezet egyik ábráján ennek a szűrőnek a működése jól látható.
A terület nálam a kísérletezés alapján *]3000,20000[* intervellumba került pixelnagyságban meghatározva.

3.4. Alakzatok

Ha sikerül felismerni a kívánt tárgyat, akkor annak képi középpontját meg kell találni. Ebből a célból megkíséreltem figyelembevenni az vezérlőnek használt tárgy alakját. A kontúrok meghatározásával ugyanis ennek megfigyelésére lehetőség is nyílik.
A detektált eszköz alakzatát belső pontjai alapján le lehet írni. Ezt momentumok teszik lehetővé. Ezekből számos olyan mérték állítható elő, melyek kielégítően jellemzik az objektumot, és az objektumon végrehajtott lineáris transzformációkkal szemben invariánsak.
Az f(x, y) 2-dimenziós folytonos függvény (p, q)-adik momentuma a valószínűség elméletéből ismert definíció alapján:

$$m_{pq} = \int_{-\infty}^{\infty} x^p y^q I(x,y) dx dy \tag{5}$$

A centrális momentum meghatározása:

$$M_{pq} = sum_x \sum_y (x - \overline{x})^p (y - \overline{y})^q I(x,y) \tag{6}$$

ahol $\overline{x} = \frac{m_1 0}{m_0 0}$ és $\overline{y} = \frac{m_0 1}{m_0 0}$

Az OpenCV *moments()* nevű függvényét használtam fel a megtalált eszközöm középpontjának meghatározásához, ez segíti elő az egér képernyőn lévő pozíciójának megváltoztatását. Az említett függvény a fenti képleteket használja fel ehhez.

3.5. Maszkolás alkalmazása

Miután a megfelelő szűrési eljárásoknak sikerült alávetni a szoftvert, alkalmazhatóvá válik a maszkolás. Látható, hogy a zajos képből hogyan lett kivehető egy egyszerű folt, amelynek a tulajdonságait figyelve kijelenthető, hogy az a keresett objektum. A maszkolás egy képösszeadás segítségével történik a következő módon:

$$J(x, y) = M(x, y)I_1 + (1 - M(x, y))I_2(x, y) \tag{7}$$

Ezt a műveletet az OpenCV *bitwise_and()* függvényével hajtódik végre, amelynek az előfeldolgozott képet át kell adni. A megfelelő pozíciók, - mivel a feldolgozott és az eredeti nyers képi bemenet egyforma méretűek - könnyen átadhatóak a másik ablaknak, ahol bekeretezésre kerül a használt eszköz.
Az eredményről átfogó ábra a következő fejezetben látható, amely már az egér vezérlésével foglalkozik.

4. Az egér vezérlése

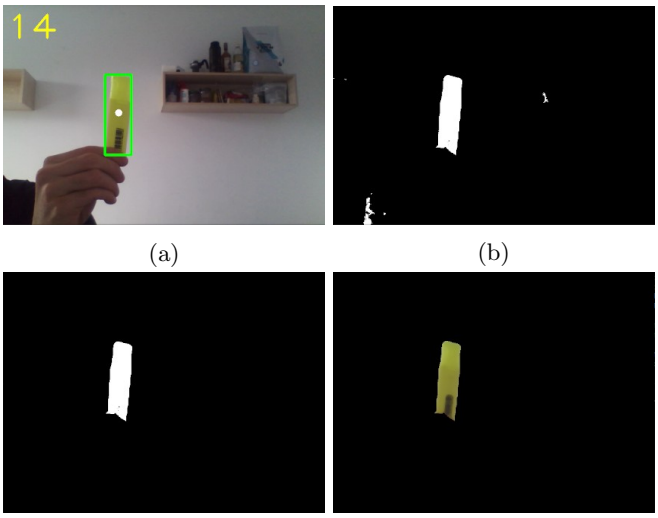
A következő fejezetben leírom az egér vezérlésével kapcsolatos megoldásaimat.

4.1. Követés középponti elmozdulás alapján

Ha sikerült megtalálni az adott objektumot, amivel a feladatot végre szeretnénk hajtani, akkor annak középpontját kell venni és a továbbiakban ennek a pontnak az elmozdulása fogja meghatározni az egér mozgását.

4.2. Az egér mozgása

A felismert eszköz középpontjának képi koordinátáit arányosítottam a képernyőm felbontásához. Ez egyszerű osztási műveletet követelt meg: Legyen a monitor felbontása wxh, ahol w a szélesség, h pedig a magasság pixelekben. Innen kiszámolható egy eltolási szorzó mind magasságra, mind szélességre a következő módon: $tX = w/cW$, illetve $tY = w/cH$, ahol cW és cH a bemeneti képet biztosító kamera szélessége és magassága.
A meghatározott centrális momentumok segítségével kiszámolható az eszköz középpontja, amely legyen a *C(cX, cY)* pont.
Ezek után meghatározható az egér pozíciója, ha vesszük, hogy $mX = cX \cdot tX$ és $mY = cY \cdot tY$, ahol mX és mY az egér koordinátái. Így az egér pozíciója az *M(mX,mY)* pont, amely a mozgás során folyamatosan változik, ezt pedig *autopy* csomagnak átadva könnyen kivitelezhetővé válik az egér mozgatása.



(a) (b)

(c) (d)

6. 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 képi bemenet ténylegesen látható részét ábrázolja.

5. A program telepítése és működtetése

5.1. 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
- autopy csomag
- screeninfo csomag az OpenCV ablakainak elhelyezéséhez
- 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.
A telepítési eljárást megkönnyítve létrehoztam egy *'requirements.txt'* állományt, amelyet könnyen fel lehet használni a szükséges csomagok telepítésére. Ehhez a következő parancsot kell lefuttatni terminál ablakban, a szoftver könyvtárába belépve:

```
python3 -m pip install -r requirements.txt
```

Ha a felhasználó egyesével szeretné letölteni a szükséges csomagokat, akkor erre is van lehetősége. Ez esetben a következő parancsokat kell lefuttatnia terminálon keresztül:

```
python3 -m pip install numpy==1.20.3
python3 -m pip install opencv-contrib-python==4.5.2.54
python3 -m pip install autopy==4.0.0
python3 -m pip install screeninfo
```

Ezek után a program futtatható lesz.

5.2. A program futtatása

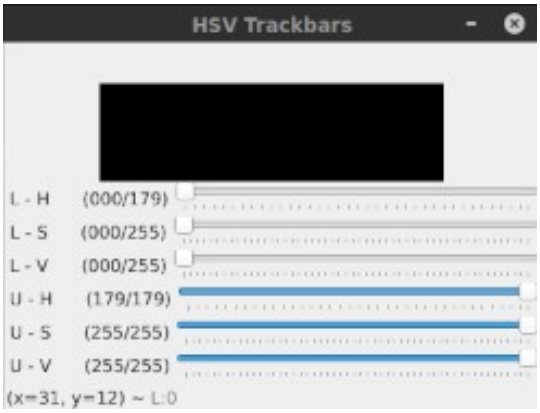
A szoftvert alapvetően kétféle módban lehet futtatni. Az első a beállítási mód, amely lehetővé teszi a felhasználó számára, hogy a HSV kódolást saját magának beállítsa, lehetővé téve számára, hogy saját eszközt is kiválaszthasson a vezérlésre. A másik mód az, amelyikben a rendszer már a megfelelő HSV-s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

5.3. HSV szűrő beállítása

A következő *–setup* kapcsolóval futtatva a szoftver egy beállítási módban indul el. A teljes futtatási parancsa a következő:

```
python3 control.py --setup
```

Ebben az üzemmódban tudja a felhasználó a középső ablak csúszkáival a H, S, V értékek alsó és felső határait beállítani és közben tudja ellenőrizni, hogy a megfelelő objektumot észleli -e a program (a felső 3 görgető az értékek alsó, az alatta lévő 3 pedig a felső határokat jelentik).



7. ábra. A HSV csúszka, amelyen beállíthatóak az alsó- és felső HSV értékek.

Ha a kiválasztott eszközt a bal felső sarokban elhelyezkedő *Camera frame* ablakban egy zöld keret veszi körül egy kis fehér körrel a közepén, és az alatta lévő ablakban is csak azt látjuk, akkor sikeresen be lettek állítva az értékek. Ezt az előző, az egér mozgásával kapcsolatos fejezetben található **ábra** szemlélteti. A '*q*' betű megnyomásával lehet kilépni, ezzel egyidőben az értékek egy külső '*.txt*' fájlba kerülnek kimentésre.

5.4. A program működése

A megfelelő beállítás után a program a következő egyszerű paranccsal futtatható:

```
python3 control.py
```

Ebben a módban már nem jelennek meg a beállításhoz szükséges csúszkák, illetve az egér is mozogni fog annak megfelelően, ahogyan a kiválasztott eszköz mozog a kamera előtt.

6. Kiértékelés

6.1. Összegzés

A dolgozatban elkészített szoftver alkalmas rá, hogy az egér vezérlését át lehessen venni akármilyen tárggyal, amely megfelel a felhasználó által megadott paramétereknek. Ezek a paraméterek a *HSV* értékhatárok. Az alakzatok meghatározásával lehetőség nyílik a középpont meghatározására is, amelyet lekövetve az egér mozgathatóvá válik.

6.2. Korlátok

A fejlesztés során az alábbi korlátozó tényezők merültek fel:

- Fontos a környezettől jól elkülönülő színű eszközt választani a feladatra. Itt kiemelném az emberi bőr színéhez hasonló színeket, mint a rózsaszín és a sárgásbarna árnyalatok. Célszerű kerülni a tiszta fehér és fekete színeket is.
- Az eszköz által határolt területet a kamerától való távolság könnyen befolyásolhatja.
- Az FPS szám korlátossága darabossá teheti az egér mozgását.
- Az egér mozgása még viszonylag könnyen megoldható, egy egyszerű kattintás viszont már nehezen kivitelezhető egy eszközzel.
- Sötét környezetben nagyon fontos tényező a fényerő, illetve a tárgyak színének telítettsége jelentős csökkenéseken mehet keresztül. Ennek hatására, hogy több eszköz megjelenése közelebb fog állni a szürkeárnyalathoz, így nehezebb őket megkülönböztetni egymástól.
- A tesztelés során felmerült egy olyan probléma is, hogy a kiválasztott eszköz viszonylag kis mozgástere miatt a képernyő szélére nem lehet elvinni az egeret.

6.3. Lehetőségek

A legutolsó pontra jelenthet megoldást egy valamivel nagyobb fokú tX, illetve tY szorzók meghatározása, viszont ez a megoldás érzékenyebbé teheti az egeret a tárgy mozgatására.
A feladat pontosabb megoldásához és a szoftver könnyebb felhasználásához inkább neurális hálókon alapuló megoldásokat érdemesebb kipróbálni. Ilyen lehet akár egy kézfelismerő rendszer fejlesztése, amelynek különböző gesztusait meg lehet figyelni. Például a *Google* által fejlesztett *mediapipe* is hasonló elven működik, illetve vannak olyan *state of the art* jellegű megoldások, amelyek segítségével már a felhasználó is betaníthat különböző neurális hálókat.

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/4.5.2/index.html (2021.11.02.)

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

[6] Hollósi János - Gépi látás előadások, Széchenyi István Egyetem, Győr, 2021

[7] Fazekas Attila, Kormos János - Digitális képfeldolgozás matematikai alapjai, mobiDIÁK könyvtár, Debreceni Egyetem, 2004