



Gépi látás
GKNB_INTM038

Hatoldalú dobókocka felismerése

ZELLES TAMÁS

GYŐR, 2019/20 I. FÉLÉV

Tartalomjegyzék

Tartalomjegyzék	2
Bevezetés	3
Feladatleírás:	3
A megvalósítás nyelve:	3
Elméleti háttér	4
Probléma felvetés:	4
A kocka értékeinek detektálása:	4
A kockák detektálása:	5
Megvalósítás terve	6
Tesztelés	8
Fekete háttér:	8
Fehér háttér:	9
Barna háttér	10
Összefoglalás	10
Felhasználói dokumentáció	11
Környezet, szükséges csomagok, programok:	11
Irodalomjegyzék	12

Bevezetés

Feladateleírás:

A feladat melyet megvalósítottam a félév során a hatoldalú dobókockával, illetve kockákkal dobott értékek felismerése. Az értékek detektálása élő képen történik, akár több kocka esetén is. A összesen dobott érték mellett a kockák számának meghatározása is megtörténik.

A megvalósítás nyelve:

A Python egy általános célú, nagyon magas szintű programozási nyelv, melyet Guido van Rossum holland programozó kezdett el fejleszteni 1989 végén, majd hozott nyilvánosságra 1991-ben. A nyelv tervezési filozófiája az olvashatóságát és a programozói munka megkönnyítését helyezi előtérbe a futási sebességgel szemben. A Python többek között a funkcionális, az objektumorientált, az imperatív és a procedurális programozási paradigmákat támogatja. Dinamikus típusokat és automatikus memóriakezelést használ, ilyen szempontból hasonlít a Scheme, Perl és Ruby nyelvekhez, emellett szigorú típusrendszerrel rendelkezik. A Python úgynevezett interpreteres nyelv, ami azt jelenti, hogy nincs különválasztva a forrás- és tárgykód, a megírt program máris futtatható, ha rendelkezünk a Python értelmezővel. A Python értelmezőt számos géptípusra és operációs rendszerre elkészítették, továbbá számtalan kiegészítő könyvtár készült hozzá, így rendkívül széles körben használhatóvá vált.[1]

Elméleti háttér

A program megvalósítás szempontjából négy részre osztható fel, az elméleti háttér esetében ennek két részéről beszélek, melyek az objektumok felismerését, illetve az értékük detektálását végzik.

Probléma felvetés:

A feladat megoldásánál több probléma merült fel, melyek megoldása szükséges volt.

Megoldandó feladatok:

- Eltérő fényviszonyok esetén történő működés
- Eltérő kocka pozíciók, illetve érték jelölési színek

A kocka értékeinek detektálása:

A kocka értékének megállapításához a folt detektálás eljárását használtam fel. A folt a képen összekapcsolt képpontok egy csoportja, melynek közös tulajdonsága van. Az esetemben a foltok nem mások mint a kocka adott oldalán található értékjelzések, és az érzékelés célja nem más mint e régiók azonosítása, megjelenítése. Az OpenCV könyvtár jó , és egyszerű megoldást nyújtanak a foltok detektálásának problémájára.

A könyvtár által biztosított megoldások közül a Simple Blob Detector-t választottam a probléma megoldására. Az algoritmus nevéből is következik, hogy egy egyszerű algoritmusról van szó, melyeket a paraméterek vezérelnek.[2]

Az alap paraméterek közé tartozik, a küszöbölés, a csoportosítás, egyesítés, valamint a középpont és a sugár számlálása. A detektor még kiegészíthető plusz paraméterekkel is. A plusz dolgok közé tartozik például a szín alapján történő szűrés, a körkörösségi ellenőrzés, mely talán a munkám során az egyik legfontosabb tényező volt, valamint még lehetőség van az algoritmusban a méret szerinti szűrésre is. A detektor segítségével megszűrt képen, ezek után a kulcspontok megkeresése, továbbá kirajzolása történik meg.

A kockák detektálása:

A kockák detektálása, az érték megállapításánál nehezebb volt. Az alapja a kockák megtalálásának az élek megtalálása a programban. A bemeneti kép szürke árnyalatossá tételével készítem elő, a Gauss féle szűrő alkalmazásával a zajokat eltávolítom a felvételtől. Az így átalakított képen az adaptív küszöbölés műveletét hajtottam végre, így lett az a kép, melyen megkeresem a kontúrokat. A megtalált összes kontúr száma, illetve megfelelő műveletek végrehajtásával, pedig meghatározom az aktuális kocka számot, amit a kamera lát.

Megvalósítás terve

Az általam megírt program több egymástól egyértelműen elkülönülő részt tartalmaz. A következőkben ezeket szeretném részletesen bemutatni.

Az első lépésben importálok a szükséges könyvtárakat. A megoldás során a numpy, mint np alkalmaztam, valamint az OpenCV könyvtárat cv2-ként. Az OpenCv használata, a leghangsúlyosabb a programban, hiszen a megfelelő észlelési, feldolgozási eljárásokat, algoritmusokat ez tartalmazza. Továbbá az XlsxWritert az adatok megjelenítésére használom külső fájlban.

A következő részben a kocka/ kockák értékeinek érzékeléséhez használatos SimpleBlobDetector algoritmust vezérlő paramétereit gyűjtöttem össze. Az oka, az, hogy ne a használati helyen kelljen mindig módosítani, illetve egyértelműen használható paraméter nevek segítenek a kód jobb értelmezésében ezen a helyen.

A következő lépésben a felvételt tároló változót inicializálok. A „felv” változóban, majd még ebben a részben a fényerőt, és egyéb szükséges paramétereket állítom be. A paraméterek a jó minőségű, fényerőjű képhez kell.

A felvétel paramétereinek beállítása után deklarálok két listát, melyek majd a dobott érték validálásához lesznek szükségesek a program későbbi részében.

A konkrét megoldás ezek után következik, egy ciklus kerül elindításra, mely mind addig fut, míg el nem éri a vizsgálatok száma, a 90000-et, vagy a SPACE billentyű lenyomásra nem kerül. Ha a feltétel érvényesül, hogy adott érték alatt van a vizsgálat száma, akkor a változók illetve a listák ismét nullázva lesznek gyakorlatilag reset lépést hajt végre a programon, ez a lépés.

A megfelelő változóba beállítom ezek után az aktuális framet, melyet majd később a megfelelő algoritmusok a vizsgálat tárgyaként tudnak használni.

A program elején deklarált paraméter értékek átadása következik utolsó lépésként a vizsgálat megkezdése előtt, itt szintén jól olvasható a program, hiszen ezt a változó nevek elősegítik.

A beállítások után történik meg konkrétan a képen lévő információ feldolgozása, konkrétan első lépésben a kocka/kockák értékeinek meghatározása. A megtalált foltokat eltárolom egy listában, mely tartalmazza az összes kulcspontot, így azokat, a következő lépésbe ki tudom rajzolni, a körvonalak megjelenítésével segítettem a elő a pontos azonosíthatóságát.

Az értékek érzékelése mellett a kockák számának érzékelése volt a feladat része. Az egyszerű feldolgozás érdekében, valamint a kocka keresésnél alkalmazott algoritmus megfelelő működése érdekében, első lépésben szürke árnyalatossá alakítom az aktuálisan vizsgált frame-t. A következő lépésben a már átalakított képen a következő lépés, amikor a felmerülő képi hibákat Gauss- féle szűrővel oldom meg, erre szükség azért volt, hogy a következő lépésekben ne detektáljon a program nem létező éleket. A konverzió után átesett képen, ezek után végre hajtok egy küszöbölést, melyet, úgy állítottam be, hogy a specifikációban lévő leírás szerinti magasságból nézve elősegítse az élek szűrését. A megejtett eljárás után, pedig meghatározom a kontúrokat. A legutolsó lépésben, pedig a kockák kontúrjainak számából meghatározom a kockák számát, melyet egy változóba tárolok.

A felismerési lépések után, az eredményt a felhasználó számára képi formában megjelenítem. A konkrét értékeket, viszont még nem írom ki a kimenetre ebben a lépésben, az egy validálási folyamatra kerül át, és azon való megfelelés után kiírom a végső értéket.

Az ellenőrzési folyamat első lépése, az hogy a megállapított értéket eltárolom egy változóba, a sok érték okán, minden tizedik frame esetén vizsgálom az aktuális értéket, és ha az utolsó három vizsgált érték megegyezik, akkor úgy veszem ez az aktuális dobott összérték.

A validált értéket, még egyszer megvizsgálom kiírás előtt, mégpedig olyan szempontból, hogy változott-e az utolsó kiírás óta az érték, vagy esetleg nulla amit lát a program, tehát nem is lát kockát, ezekben az esetekben a sztenderd kimenetre nem jelzem az értékeket, ha megfelel az elvárásnak akkor, dobott érték: kocka szám formában kiíratom a kimenetre.

Befejezés képpen a ciklus az újabb vizsgálat előtt növeli a számláló értékét, valamint, figyel a kilépési feltételre, hogy lenyomása kerül-e a SPACE billentyű. Majd, ezek után egy „eredmeny.xlsx” nevű és kiterjesztésű fájl megfelelő soraiba kerülnek kiírásra, a könnyebb értelmezhetőség okán.

Tesztelés

A program megfelelő működését tesztek segítségével ellenőriztem. A teszteket a kocka több állásában, a kockák több elrendezésében hajtottam végre, valamint eltérő homogén, illetve inhomogén felületek felett. A különböző hátterek és megvilágítás esetén 50-50 tesztet hajtottam végre. A tesztek során a kamera a vizsgált terület felett 21cm-re volt a terület síkjával párhuzamosan elhelyezve.

Fekete háttér:

Fekete háttérnél nagyobb működési pontosságot fedeztem fel, a kockák közelsége sem volt zavaró, 0,5 cm távolság két kocka között sem okozott problémát a programnak működése során.

33,5 lx fényerő esetén:

50 tesztből átlagosan 13 esetben kaptam hibás eredményt, még 37 esetben pontos eredményt, ez azt jelenti, hogy 74%-ban helyesen azonosított a program, 26%-ban nem. A hibás eredmények 12 esetben a kocka/kockák számának meghatározásánál, míg 1 esetben az érték meghatározás volt hibás átlagosan. A hibákból továbbá szembetűnő, hogy a nagyobb dobott értékek esetén téved a program, valamint, ha a kép széléhez közel található meg a kockák valamelyike, vagy mindegyik.

A teszt eredményeket tartalmazó Excel tábla a Github repoban:

https://github.com/ztamas97/Gepilatas-GKNB_INTM038-/blob/master/fekete33%2C5lx.xlsx

Teszt adatbázis:

https://github.com/ztamas97/Gepilatas-GKNB_INTM038-/blob/master/fekete33%2C5lx.rar

10,4 lx fényerő esetén:

50 tesztből átlagosan 16 esetben kaptam hibás eredményt, még 34 esetben pontos eredményt, ez azt jelenti, hogy 68%-ban helyesen azonosított a program, 32%-ban nem. A hibás eredmények 14 esetben a kocka/kockák számának meghatározásánál, míg 2 esetben az érték meghatározás volt hibás átlagosan. A program futásánál érezhető, hogy a kép széléhez közel álló kockák meghatározása működik kevésbé.

A teszt eredményeket tartalmazó Excel tábla a Github repoban:

https://github.com/ztamas97/Gepilatas-GKNB_INTM038-/blob/master/fekete10%2C4lx.xlsx

Teszt adatbázis:

https://github.com/ztamas97/Gepilatas-GKNB_INTM038-/blob/master/fekete10%2C4lx.rar

Fehér háttér:

Fehér háttér esetében két kocka között szükséges volt az elfogadható működéshez a minimum 1cm-es távolságra két kocka között. Viszont ezek után sem kaptam megfelelő eredményt az 50 tesztből sajnos minimális számú volt a pozitív eredmény, hiszen nem voltak megfelelőek a kontúrok. Ennek oka, hogy a tesztek során fehér alapon fekete jelzésű kockákat használtam, és azok nem megfelelően ütnek el a fehér háttértől. Természetesen helyes detektálások itt is történtek, de különböző fényviszonyok között is a legjobb arány 15 helyes felismerés volt 50 tesztből.

33,5 lx fényerő esetén:

50 tesztből átlagosan 38 esetben kaptam hibás eredményt, még 12 esetben pontos eredményt, ez azt jelenti, hogy 24%-ban helyesen azonosított a program, 76%-ban nem. A hibás eredmények 38 esetben a kocka/kockák számának meghatározásánál volt a probléma, a kockák értékeinek meghatározása hibátlanul működött. A hibákból továbbá szembetűnő, hogy a probléma ebben az esetben egyértelműen a nagyobb kocka számnál, és a nagyobb dobott értéknél jelentkezett.

A teszt eredményeket tartalmazó Excel tábla a Github repóban:

https://github.com/ztamas97/Gepilatas-GKNB_INTM038-/blob/master/feh%C3%A9r33%2C5lx.xlsx

Teszt adatbázis:

https://github.com/ztamas97/Gepilatas-GKNB_INTM038-/blob/master/feh%C3%A9r33%2C5lx.rar

10,4 lx fényerő esetén:

50 tesztből átlagosan 35 esetben kaptam hibás eredményt, még 15 esetben pontos eredményt, ez azt jelenti, hogy 30%-ban helyesen azonosított a program, 70%-ban nem. A hibás eredmények 12 esetben a kocka/kockák számának meghatározásánál, míg 3 esetben az érték meghatározás volt hibás átlagosan. A program futása során egyértelműen érzékelhető volt, hogy a kisebb kocka szám esetén volt helyes leginkább a program működése.

A teszt eredményeket tartalmazó Excel tábla a Github repóban:

https://github.com/ztamas97/Gepilatas-GKNB_INTM038-/blob/master/feh%C3%A9r10%2C4lx.xlsx

Teszt adatbázis:

https://github.com/ztamas97/Gepilatas-GKNB_INTM038-/blob/master/feh%C3%A9r10%2C4lx.rar

Barna háttér:

Annak okán, hogy ne csak a fekete illetve fehér felületek előtt legyen tesztelve a program, eltérő felülettel is teszteket hajtottam végre. A fényerősség ebben az esetben 20,5lx volt egységesen a tesztek lefuttatásánál. Az eredmények azt mutatták, hogy 50 dobásból 32 esetben történt sikeres azonosítás, még 18 esetben hibás eredményt kaptam. A esetek 64%-ban sikeres, a tesztelési folyamat 36%-ban pedig helytelen eredményt kaptam.

A teszt eredményeket tartalmazó Excel tábla a Github repoban:

https://github.com/ztamas97/Gepilatas-GKNB_INTM038-/blob/master/barna20%2C5lx.xlsx

Teszt adatbázis:

https://github.com/ztamas97/Gepilatas-GKNB_INTM038-/blob/master/barna20%2C5lx.rar

Összefoglalás

A lefuttatott tesztek eredményeit az alábbi táblázatban foglaltam össze.

Háttér színe	Fényerősség (lx)	Tesztek száma	Sikeres az.	Sikertelen az.	Sikeres az. %	Sikertelen az. %
fekete	33,5	50	37	13	74	26
fekete	10,4	50	34	16	68	32
fehér	33,5	50	12	38	24	76
fehér	10,4	50	15	35	30	70
barna	20,5	50	32	18	64	36

A tesztenként, illetve képenként lebontott eredményeket az adott körülmények taglásánál található linken lévő Excel táblában találhatóak meg. A teszt adatbázisok tömörített állományban szintén a feladat repojában található meg.

Felhasználói dokumentáció

Környezet, szükséges csomagok, programok:

- Működési/tesztelési környezet:
 - HW: Asus Vivobook
 - OS: Windows 10 Home
- Fejlesztői környezet:
 - OS: Windows 10 Home
 - IDE: Thonny 3.1.2
- A szükséges csomagok a program működéséhez:
 - Numpy
 - OpenCV
 - XlsxWriter

A program forráskódja, valamint dokumentációjának elérhetősége:
https://github.com/ztamas97/Gepilatas-GKNB_INTM038-

A program fejlesztése során webkamerát használtam a kockák megfigyelésére. Az érzékelő a kockák dobási síkjával párhuzamosan, attól 21cm-re volt rögzítve a teljes tesztelési folyamat során. A kockák fehér alapon fekete ponttal történő értékjelzésűek voltak, az oldal hossz egységesen 1,5cm-es. A háttér homogén fekete, mely során a tesztek jó eredményt adtak.

A program indítása után a kockák az érzékelő elé helyezve az érték rövid időn belül kijelzésre, valamint kiírásra kerül. A maximum kocka szám a három melyre a program tesztelve lett. A kockák közötti minimális távolság 1 cm, és a kép szélétől minimum 0,5 cm-re kell lennie a kockának, a megfelelő működéséhez. A programból való kilépéshez a SPACE billentyűt kell lenyomnia s felhasználónak. A program ezek után leállítja a kamerát is.

Irodalomjegyzék

- [1] [https://hu.wikipedia.org/wiki/Python_\(programoz%C3%A1si_nyelv\)](https://hu.wikipedia.org/wiki/Python_(programoz%C3%A1si_nyelv))
- [2] <https://www.learnopencv.com/blob-detection-using-opencv-python-c/>