Gépi látás

GKNB_INTM038

Hatoldalú dobókocka felismerése

ZELLES TAMÁS

GYŐR, 2019/20 I. FÉLÉV

Tartalomjegyzék

Bevezetés

Feladatleírás:

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 mellet 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 felvételről. 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 importálom 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 inicializálom. 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 mellet 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 framet. 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 konverzión 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 érétket.

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 kiíratás elött, mégpedig olyan szempontból, hogy változott, e az utolsó kiíratás 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 elött 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 kiíratásra, 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.

50 tesztből átlagosan 13 esetben kaptam hibás eredményt, még 37 esetben pontos eredményt. A hibás eredmények 12 esetben a kocka/kockák számának meghatározásánál, míg 1 esetben az érték meghatározás volt hibás átlagosan.

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.

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)

[2] https://www.learnopencv.com/blob-detection-using-opencv-python-c/