Galgóczi Áron

NBII6E

Gépi látás - (GKNB_INTM038)

2020.12.16.

Tartalomjegyzék

Dobókocka számláló alkalmazás

1 Feladat

Bármilyen fényviszony és háttér előtt képes legyen megszámlálni maximum 4 db dobókocka összértékét illetve a dobókockák darabszámát is jelezze ki.

2 Elméleti háttér

- Interpoláció során a képeket nagyítjuk (x > 1) vagy kicsinyítjük (1 < x < 0) ahol x az interpoláció mértéke. A folyamat során a kép nem ismert értékeit közelítjük az ismert értékek felhasználásával.
- Medián szűrő: A vizsgált kép egy egységnyi méretű részlete alapján képzett számsorozat mediánja lesz az új intenzitás érték. Só bors jellegű zajok eltüntetése, de jelen feladatban a felesleges éllek eltávolítását végzi.
- Morfológia: Képek összehasonlítása egy csúsztatott strukturáló elemmel. Ennek hatására bináris képeken alakzatokat lehet kiemelni a vonal rengetegből melyek így egynek látsszak. Valamint szürke árnyalatos képen képes kihangsúlyozni a kontúrokat egy mediánnal szűrt képen.
- Kör detektálás: két eljárás van. Az egyik az ismert méretű köröket keres, a másik ismereten méretűeket. A lényeg, hogy a bináris kép pontjain egy vagy több kört helyez el. Ahol ezeknek a köröknek az íve metszi egymást ott potenciális kör középpont található, melynek a sugara előre definiált vagy a metsző körök átlaga.
- Él detektálás: kiemeli azokat a helyeket, ahol az intenzitás nagy mértékben változik.

3 Megvalósítás terve és kivitelezés

- Terv: kockák pontjai körök így azokon lehet kör keresést végrehajtani, valamint egy kellően homályos képen egy kocka szintén kör alakúnak látszik.
- Megvalósítás: 3 különböző fájl:
  - Viszgalat.py: Itt van definiálva a vizsgal(dir,bool) függvény. A függvény meghívásakor egy elérési utat tartalmazó szöveget vár és egy logikai változót vár, mely felelős engedélyezni a kép megjelenítését.
    Az elején probléma adódik a képek felbontásával, mert egy 3000 pixel szélles képen egy kocka pl. 100 pixel szélles, de egy kisebb képen ez jelentősen csökken, így mindenképp szükség van átalakításra.
    - Pont detektálás: a képeket 2000÷szélllesség arány számmal interpoláltam így egy nagyobb kép esetén is 50 pixel maximum lehet egy pötty. Ezután 8-bites szürke árnyalatos képpé alakítom. majd ami át esik egy medián szűrön és morfológiai záráson, és ezután következik a körök detektálása, majd az eredmények képre rajzolása ahol egyben számolja is a dobás értékeit. »korok« nevű változó tartalmazza a dobás értéket
    - Kocka detektálás: Ezt egy 500 pixel szélességű képen hajtom végre. Szintén egy 8-bites szürke árnyalatos képen melyet követ egy morfológia zárás és egy medián szűrés. A kis méretű kép előnye, hogy jobban elmosódottabb lesz a kép és folyik egybe mint pl. egy medián szűrés esetén. Az így kapott képen a kockák kör szerűbbek. »kokca« nevű változó tartalmazza a kockák számát.

A következő rész az eredmény írja ki a képre a »cv2.putText()« függvény segítségével és jeleníti meg a »megjelenit« változó bool értékének megfelelően, itt van lehetőség a kilépést biztosító »q« betű lenyomására.

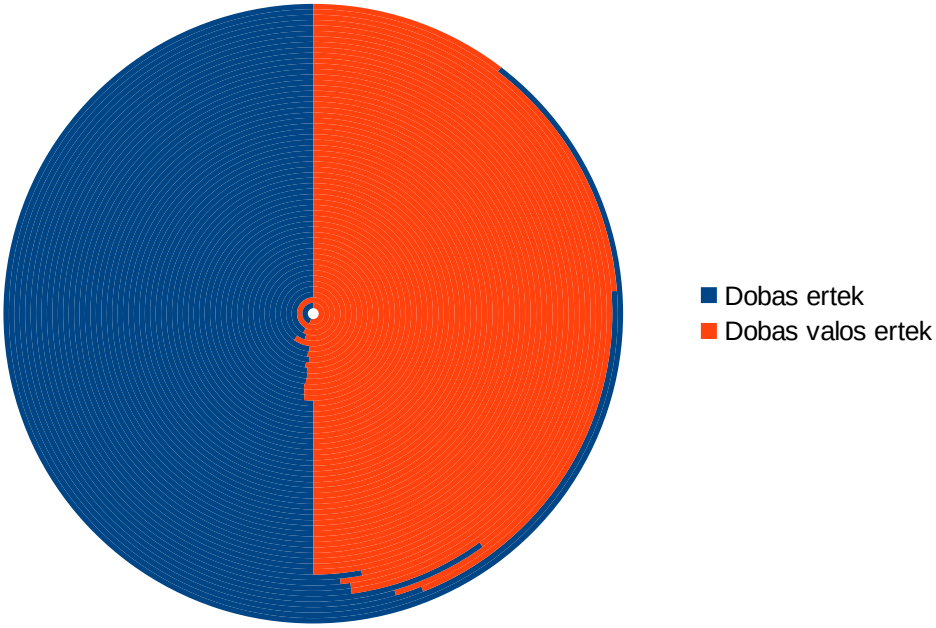
A futás végén egy tömbben vissza adja a kép nevét, a pontok és kockák számát és visszaad egy nullát, ha a nem engedélyezik a kép megjelenítést, különben a cv2.waitKey(0)- kor leütött billentyű kódját mely ha »q« kódját tartalmazza a program megszakítja a futását.

- kockadobas.py: A felhasználónak készült program. Az első sorban lehetőség mappa név megadására. Importálja a vizsgal függvény a vizsgalat fájlból, a munpy-t, cv2-t és az os- t. A program kér egy útvonalat és át adja a »megjeleníti«-nek, ha nem kap akkor végig iterál a ./képek mappán és egyesével megjeleníti a képeket.
  A »megjelenit« nevű függvény kap egy elérési útvonalat és ellenőrzi cv2.haevImageReader eljárás segítségével, hogy képről van-e szó, hiba esetén kiírja az útvonalat és egy üzenetet: „Ez nem kép!”. Különben az »eredmeny« váltózóban tárolja a »vizsgal« függvénytől kapottakat és (-1)-gyel tér vissza kilépés esetén.
- teszt.py: A teszteléshez készült program. A vizsgalat.py-ból a vizsgal függvényt, a numpy-t, a cv2-t ést az os-t importájla. A program az elején megkérdezi a tesztelőt, hogy vannak-e új képek az adatbázisban és ezt a »uj« nevű vátltózóban tárolja el, ezután „Eredmeny_mert.txt” fájl nyitással kezd annak függvényében hogy vannak-e »uj« képek mert akkor ezen felül megnyitja „Eredmenyek_valos.txt”-t is, és felírja a következő oszlopokat, akár mindkét fájlban is:
  - Kép neve
  - Dobás értéke
  - kockák szám

  Ezután az előre definiált mappa tartalmát olvassa és ellenőrzi a cv2.haevImageReader eljárás segítségével, hogy képről van-e szó, hiba esetén kiírja az útvonalat és egy üzenetet: „Ez nem kép!”. Különben az »eredmeny« váltózóban tárolja a »vizsgal(kep,megjelenjen)« (ha a »megjelenjen«-t 1-re átírjuk a kódban megjelenek a képek egyesével) függvénytől kapottakat, majd ezeket írja fájlba, ha van »uj« kép akkor bekéri a képen látható pontok és kockák számát és ezeket is a megfelelő fájlba írja. A végén ellenőrzi a tesztelő kilépési szándékát.

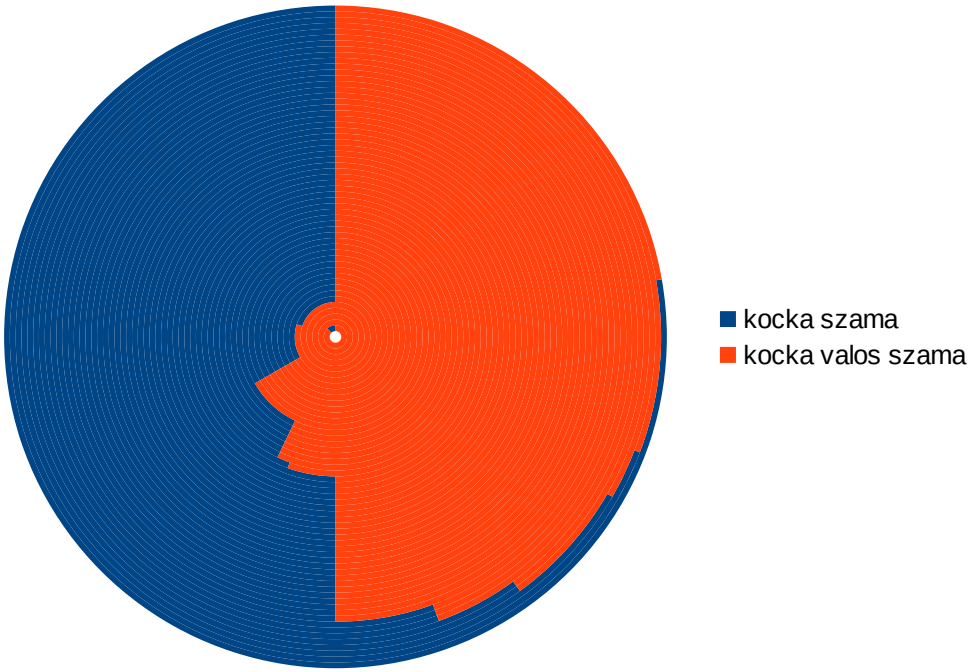
4 Tesztelés

A tesztelés az eredményeket a „Eredmeny_mert.txt”, „Eredmenyek_valos.txt” szöveges dokumentumokból táblázat kezelő program segítségével lehet hasonlítani. Ehhez először a teszt.py futtatása szükséges bármely olyan grafikus felülettel rendelkező operációs rendszeren, melyen telepítve van a python és a numpy és cv2 modulok. Ha kép adatbázison nem módosítunk *(FONTOS: a program nem tudja eldönteni, hogy tényleg módosítottunk-e vagy sem! Ez teljesen a tesztelő felelőssége)* lehetőségünk van a háttérben lefuttatni a tesztelést ehhez a »megjelenjen« változót kell 0-ra állítani ezt csak az »uj« változó képes ezt felülírni. A dobott értéket a valós értékből kivonva, csökkenő sorba rendezve, látszik a változtatások utáni különbségek:



A változás nyomon követése a diagram arány változásai tükrözik, a középső alsó közel egybe fűggő „függőleges” vonal jelzi a pontosságot. Ez számszerűen 56.1% teljes pontosság és ± 1 hiba esetén 80,7% .

Sajnos a kockák darab számát már nehezebb ez alapján javítani, mert a középső sáv nyújtása nehézkes. Első sorban az alacsony értékek miatt. A kockák számát a valós számból kivonva, csökkenő sorba rendezve, látszik a változtatások utáni különbségek, a pontosság: 43,9%, ±1 hiba esetén 63.16%.



5 Felhasználói leírás

A program futtatására a Thonny nevű programot ajánlom, de megteszi bármely Python értelmező melyet grafikus felületről indítunk.

Én Thonnyval való használatot fogom bemutatni. A thonny első indítása után a Tools menüpontban az Open system shell lehetőséget választva, adjuk ki a következő

parancsot:

- pip install opencv-contrib-python

Néha a pip helyet pip3 kell írni.

Bármi további előtt töltsük le https://github.com/Kovis97/Gepi-latas oldalról a kockadobas.py és vizsgalat.py nevű fájlokat. Mozgassuk a kívánt mappába. Ezután a Files menüpontban az open lehetőgén keresztül nyissuk meg a programot nyissuk meg kockadobas.py programot. Az első sorban módosítsuk a mappa elérési útvonalat, ha ha több képet szeretnénk vizsgálni. A futtatáshoz nyomjuk F5-öt. A program megkérdezi, hogy egy képet szeretnénk-e feldolgozni, ebben az esetben írjuk be a kép útvonalát (Pl.: C:\képek\kép.jpg), amenyiben rosszul adtuk meg, vagy nem képre mutat a program erre figyelmeztetni fog és megáll. („Ez nem kép!”) Ha több képet szeretnénk vizsgálni akkor hagyjuk üresen, ekkor a kockadobas.py nevű program első sorába idéző jelek közé írjuk be a mappa elérési útvonalát, ha rosszul adtuk meg, vagy nem csak képek vannak benne a program ekkor is „Ez nem kép!” üzenettel reagál. Kép megjelenése után billentyű nyomásra leáll vagy a következő képre ugrik a program, de „q” betűre letudjuk állítani.

6 Irodalomjegyzék

- http://www.inf.u-szeged.hu/~tanacs/pyocv/krk_detektlsa.html
- https://docs.opencv.org/master/d4/da8/group__imgcodecs.html
- https://hup.hu/comment/654902#comment-654902
- https://stackoverflow.com/questions/2349991/how-to-import-other-python-files
- https://infopy.eet.bme.hu/fajlkezeles/