Galgóczi Áron
NBII6E

Gépi látás - (GKNB_INTM038)

2020.12.22.

Tartalomjegyzék

Dobókocka számláló alkalmazás

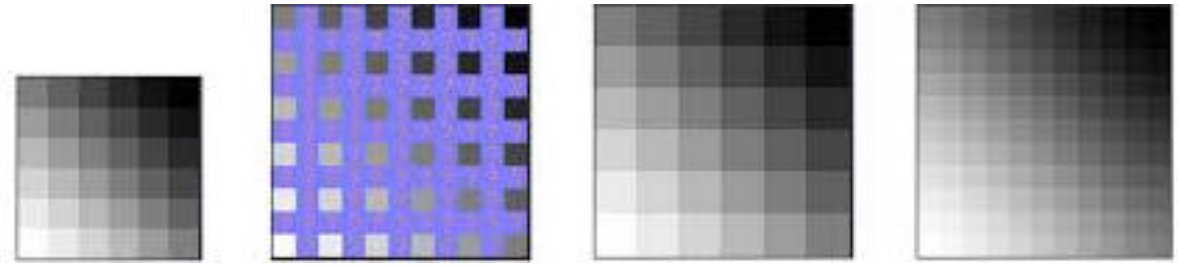
1. Feladat

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

2. Elméleti háttér

Transzformációs problémák kiküszöbölhetőek interpolációs technikákkal. Az 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. Három alapvető technika létezik az interpolációra:

- Legközelebbi szomszéd elve: A 4 legközelebbi szomszéd közül annak az értékét vesszük, amelyik a legközelebb van
- Bilineáris interpoláció: A 4 legközelebbi szomszéd távolság függvényében súlyozott átlaga
- Bicubic interpoláció: A 16 legközelebbi szomszéd alapján parciális deriválás felhasználásával



Eredeti Hiányzó részek probléma Legközelebbi szomszéd Bilineáris és bicubic

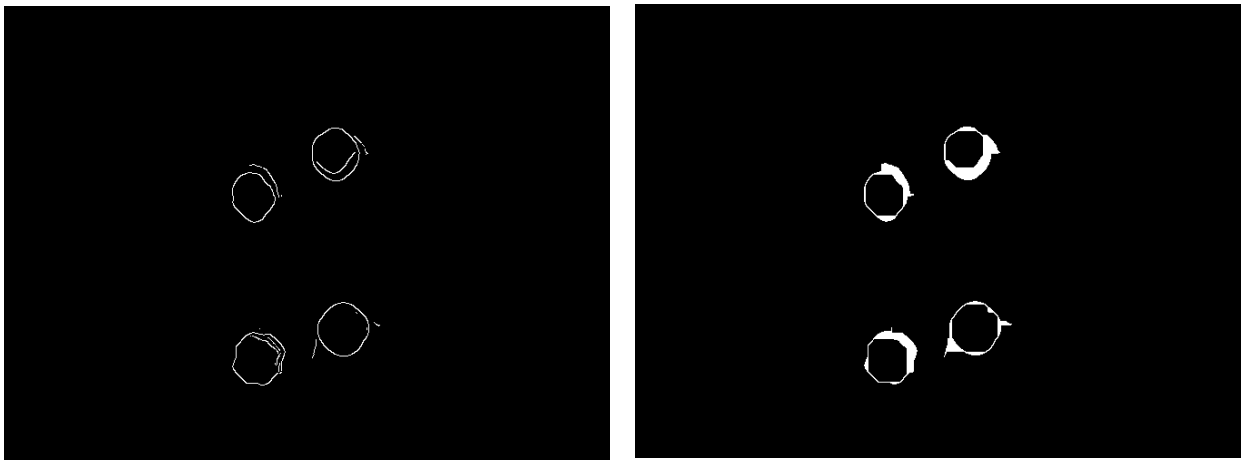
A medián szűrő folyamata során a vizsgált kép egy egységnyi méretű részlete alapján képzett számsorozat mediánja lesz az új intenzitás érték.

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

Só bors jellegű zajok eltüntetésére használják, de jelen feladatban a felesleges éllek eltávolítását végzi.

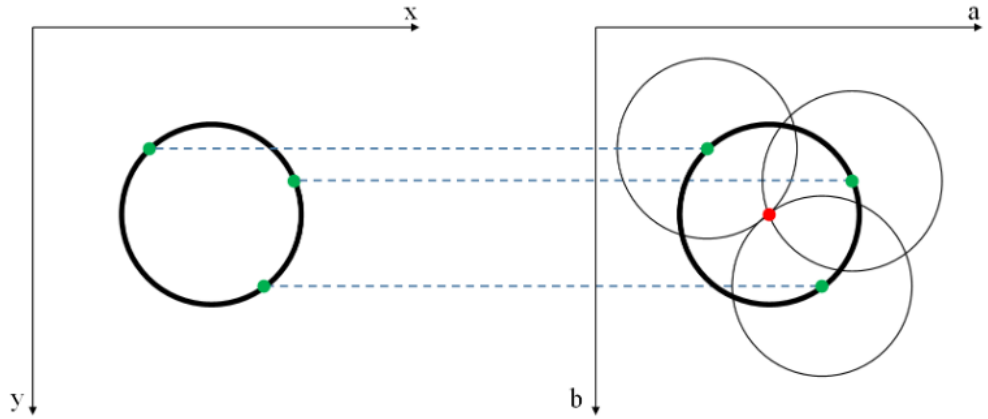
1. ábra Eredeti kép és mediánnal szűrt kép

A morfológia a képek összehasonlítása egy csúsztatott strukturáló elemmel. Ennek hatására bináris képeken alakzatokat lehet kiemelni egy általunk definiált négyzetes kernel segítségével.



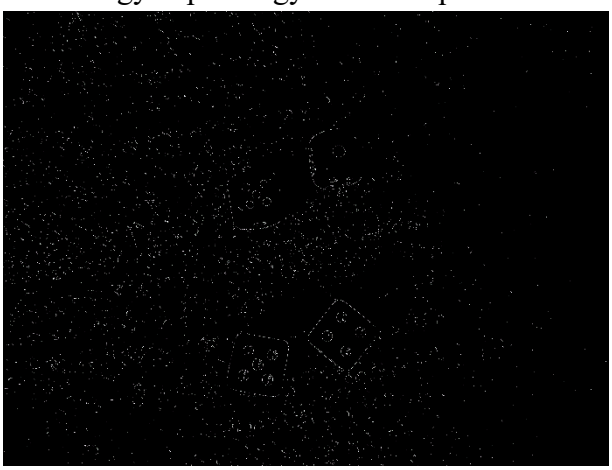
2. ábra 1.ábra mediánnal szűrt képe canny detektorral és a morfológiai zárás

A kör detektálásra két külön megközelítés van. Az egyik az ismert méretű köröket keres, a másik ismeretlen méretűeket. A lényeg, hogy a bináris kép pontjain egy vagy több változó sugarú 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 sugarának átlaga.



3. ábra dia6.pdf 34 oldala

Él detektálás során megjelöljük azokat a helyeket, ahol az intenzitás nagy mértékben változik így kapunk egy bináris képet.



$$f'(x) \approx \frac{f(x+h) - f(x-h)}{h}$$

4. ábra Él detektált kép

3. Megvalósítás terve és kivitelezés

Terv szerint a kockák pontjai körök így azokon lehet kör keresést végrehajtani, valamint egy kellően homályos képen egy kocka szintén kör alakúnak látszik.

Megvalósítás 3 különböző fájlban, egy tartalmazza a detektálást és a maradék kettő egy tesztelői és egy felhasználói alkalmazás, melyek az első meghívásért felelnek.

1. 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 adódott a képek felbontásával, mert egy 4000 pixel széles képen egy kocka pontja pl. 80 pixel átmérőjű, de egy kisebb képen ez jelentősen csökken, így mindenképp szükség van átalakításra így a képeket 2000÷széllesség arányszámmal interpoláltam így egy nagyobb kép esetén is 50 pixel lehet maximum egy pötty.

5. ábra 2000 pixel szelességűre átalakított kép

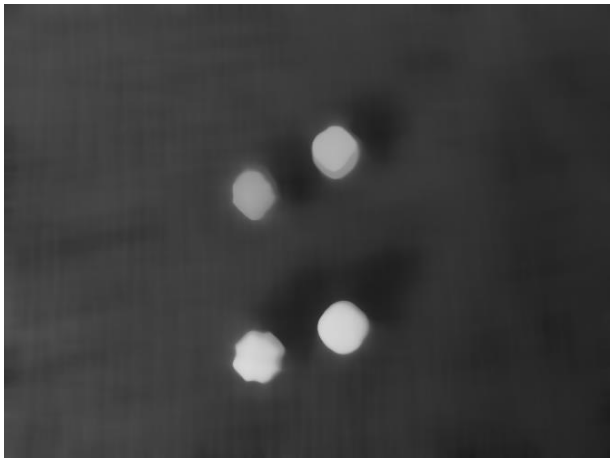
Ezután 8-bites szürke árnyalatos képpé alakítom és alkalmazok rajta egy medián szűrőt és morfológiai zárást.

6. ábra Medián morfológiai zárás

Következő lépés a körök detektálása, a cv2 HoughCircles függvényével, majd a kapott pontok képre rajzolása és egyben azok számolása is. »korok« nevű változó tartalmazza a dobás értéket.

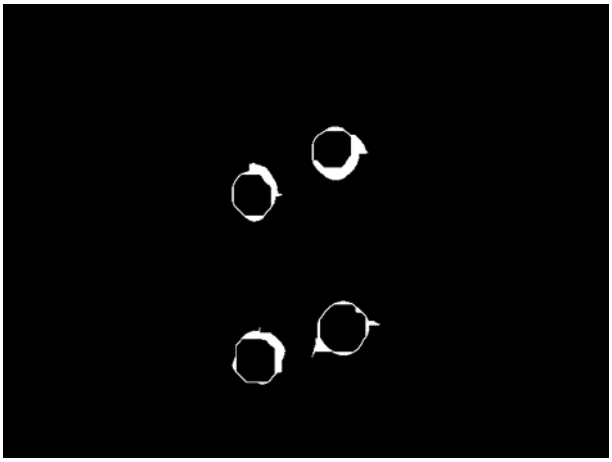
7. ábra Detektált körök a képen

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.



8. ábra 500 pixel széllas kép medián szűréssel

A kis méretű kép előnye, hogy medián szűrésnél kisebb kernel méretet kell alkalmazni, ezzel gyorsítva a programot. Az így kapott képen a kockák körszerűbbek. Ugyan a kör kereső algoritmus rendelkezik medián és élkereséssel, de az nem annyira hatékony, ezért azt külön végre hajtottam egy morfológiai zárással együtt.



9. ábra Ezzen kerresük kockákat

Az eredmény rajzolása közben »kocka« nevű változó számolja a kockák számát. A

következő rész a dobás értéket és a kockák számát 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.

10. ábra A felrajzolt kockák és pontok valamint ezek db számai

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

2. kockadobas.py a felhasználónak készült program. A program első sorában lehetőség van egy mappa név megadására, hogy mely mappából olvassunk több képet. Ezután importálja a vizsgal függvényt a vizsgalat fájlból, a munpy-t, cv2-t és az os- t. A futás során a program kér egy útvonalat, melynek egy képre kell mutatnia és át adja a »megjelenit« -nek, ha üresen hagyjuk akkor végig iterál az első sorban megadott mappán és egyesével megjeleníti a benne lévő képeket. A »megjelenit« nevű függvény a kapot elérési útvonalat ellenőrzi a cv2.haveImageReader eljárás segítségével, hogy képről van-e szó, hiba esetén kiírja az útvonalat és egy üzenetet: „Ez nem kép!”. Különben az »eredmeny« változóban tárolja a »vizsgal« függvénytől kapottakat és (-1) -gyel tér vissza kilépés esetén.

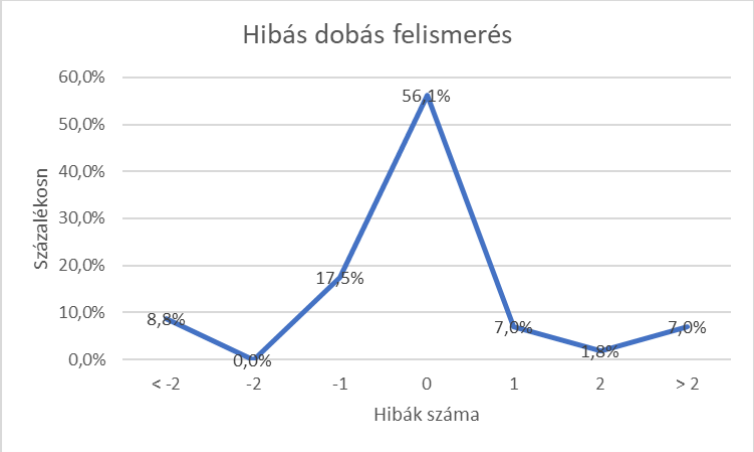
3. teszt.py: A teszteléshez készült program. A vizsgalat.py-ból a vizsgal függvényt, a numpy-t, a cv2-t ést az os-t importálja. A program az elején megkérdezi a tesztelőt, hogy „Vannak-e új képek? I/H” az adatbázisban, I/i az igazat jelenti, bármely más a hamisat jelenti, ezt az »uj« nevű változóban tárolja el. Ezután az „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 és Kockák száma. Ezután az előre definiált mappa tartalmát olvassa és ellenőrzi a cv2.haveImageReader eljárás segítségével, hogy képről van-e szó, hiba esetén kiírja az útvonalat és egy üzenetet: „Ez nem kép!”. Különben az »eredmeny« változóban 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. Az eredmény fájlok tartalmazzák a kép nevét, a pontok számát és a kockák számát is.

4. Tesztelés

A tesztelés során az eredményeket a „Eredmeny_mert.txt”, „Eredmenyek_valos.txt”

szöveges dokumentumokból táblázat kezelő program segítségével lehet kinyerni. Ehhez először a teszt.py futtatása szükséges a „Vannak-e uj kepek?” kérdésre I vagy i válaszolva, bármely olyan grafikus felülettel rendelkező operációs rendszeren, melyen telepítve van a python, a numpy és cv2 modulok. Ha a kép adatbázison nem módosítunk *(FONTOS: a program nem tudja eldönteni, hogy tényleg módosítottunk-e vagy sem! Ez teljesen a tesztelő felelőssége)* lehetőségünk van a képek nem megjelenítésével gyorsítani a tesztelést, ehhez a »megjelenjen« változó a jelzett sorban 0-ra kell állítani, ezt csak az »uj« változó képes csak felülírni.

A program a kocka dobás felismerésének pontossága az alábbi ábrán látható:



11. ábra Kocka dobás hiba szám százalékosan

Az eredmények tükrében elmondható, hogy ezen beállítások mentén rétre hibák jövő zöme orvosolható lenne, de akkor az más képeken jelentene problémát.

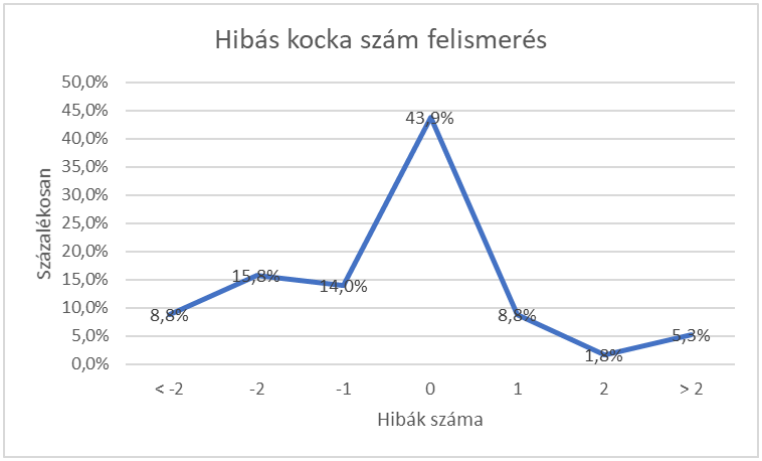
pl.:

12. ábra 56.jpg

Ezen kép több különböző színárnyalatú kocka látható, 91 ponttal, melyek közül több nem teljesen kör alakúl. Ha ezen kép alapján paraméterezem a programot akkor is 82 pontot ismer fel, viszont az egész adatbázisra nézve már csak közel 40%- ban végez helyes detektálást.

Tehát a program akkor működik megbízhatóan, ha kockák pontja épek, a kocka és pontjai között erős a kontraszt, nem verik vissza környezetük fényét és kb. 20 cm-re vannak merőlegen a kamera lencséjére, ezzel elkerülve a túl kicsi és túl nagy pontokat, valamint a kocka oldalát

A program kocka felismerés pontossága az alábbi ábrán látható:



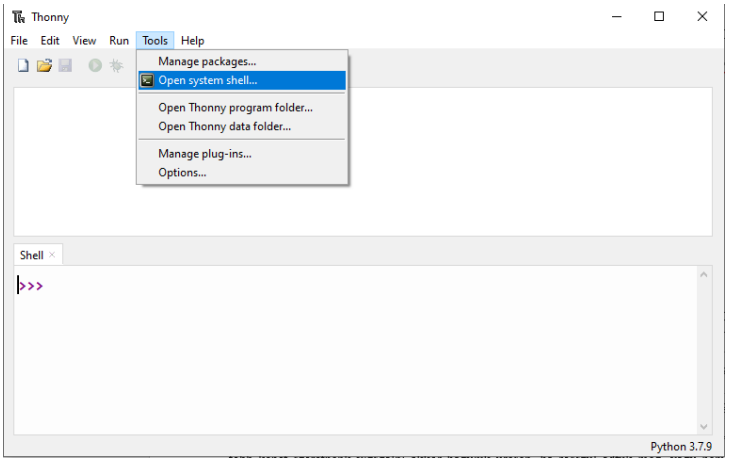
13. ábra Kockák hibás felismerése százalékosan

Elmondható, hogy akkor a legpontosabb a felismerés, ha a kocka és háttere között érezhető kontraszt van. ha nem látszik a kocka árnyéka és nincsenek világos foltok a képen pl. a vaku miatt.

5. Felhasználói leírás

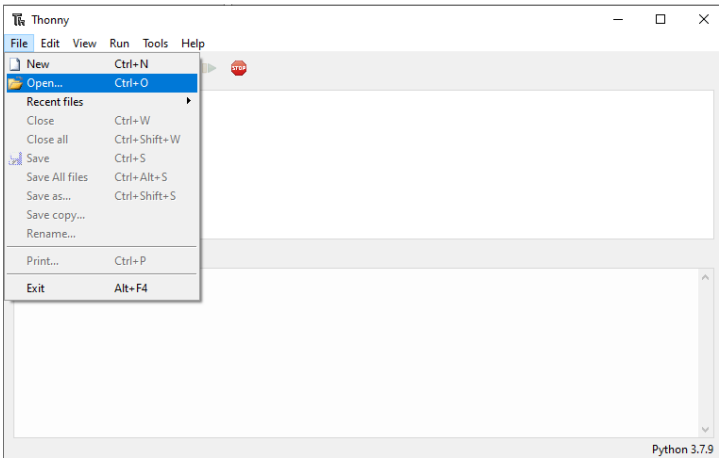
A program futtatására a Thonny nevű programot ajánlom, de megteszi bármely Python értelmező melyet grafikus felületről indítunk.

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



14. ábra Thonny program első lépései

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 lehetőségen keresztül nyissuk meg a kockadobas.py programot.

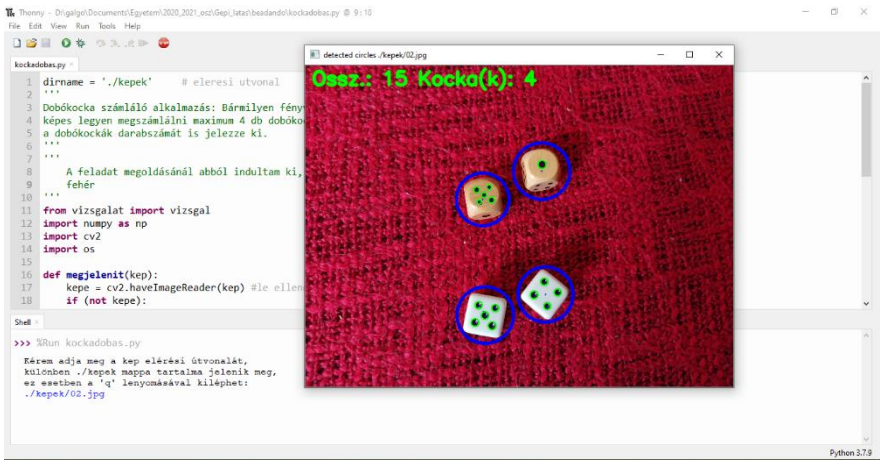


15. ábra Fájl megnyitás

Az első sorban módosíthatjuk a mappa elérési útvonalat, ha több képet szeretnénk egymás után vizsgálni. A futtatáshoz nyomjuk F5-öt.

16. ábra betöltött fájl előnézet

A program megkérdezi, hogy egy képet szeretnénk-e feldolgozni, ebben az esetben írjuk be a kép útvonalát (Pl.: C:\képek\kép.jpg), amennyiben 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, ha rosszul adtuk meg, vagy nem csak képek vannak benne a program ekkor is „Ez nem kép!” üzenettel reagál. Egy kép megjelenése után egy billentyű lenyomására leáll vagy a következő képre ugrik a program, ekkor „q” betűre letudjuk állítani. A dobott kockák száma és azok összértéke a képre kerül kiírásra.



17. ábra A program kérdése egy lehetséges válasszal és a program kimente

A találat javítása érdekében célszerű olyan képet adni a programnak mely nagy felbontású (4000*3000) és nincsenek erős árnyékok, valamint fény visszaverődés sem kockáról, sem a háttérről.

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/