

## Fejlesztői dokumentáció

Szoftveres követelmények:

A környezet amire szükségünk lesz:

Python 3.6 vagy újjab

Python 3 modulok:

- scikit-image modul
- imutils modul
- opencv-python modul
- numpy modul
- os modul
- argparse modul

A scan.py kód a git repóból :

<https://github.com/N7Remus/CV/blob/master/python/scan.py>

Tesztelve lett Manjaro linuxon, Ubuntu LTS 18.04 linuxon-en, és Windows 10-en.

- Linux alatt - telepítsük a csomagkezelőből a python3 csomagot, illetve a pip3 csomagot, majd telepítsük a modulokat is.
- Windows alatt - válasszunk egy python környezetet, majd telepítsük (a modulokat is).

Teszteléshez én a PyCharm programot ajánlom, ami könnyíti a modul letöltést.

<https://www.jetbrains.com/pycharm/>

Windows specifikusan a Visual Studio is remek alternatíva lehet, mivel van neki egy kiváló python csomagja

<https://visualstudio.microsoft.com/>

Az egyszerűség kedvelőinek a Thonny python editort ajánlom:

<https://thonny.org/>

Webes rész:

-----  
Kipróbálható a következő URL-en :

<http://remus.hungaroprofil.hu/rgabor.php>

Funkciója : a program funkciójának bemutatása, a program használatának segítése.

- -letöltés nélkül
- -telepítés nélkül
- -technika tudás nélkül

Felépítése illetve technikai leírás:

A webes rész PHP7.x-et használ, de egyszerűsége miatt visszafele is kompatibilis.

A webszerver Apache 2.4 alatt fut.

A fájlok elhelyezése tetszőlegesen történik, viszont a python fileok elérését, illetve a kész fájlok elérésének útvonalát meg kell adni a php-nak. (szerver oldali config)

A fileok elérési útvjai:

```
python script : /srv/cv_core/virtualenvironment
python kimenet : /srv/cv_core/uploads
php script : /var/www/html
```

## Működés

A php script átveszi a képet, ellenőrzi, hogy valóban kép került e átadásra. Ezt követően lementi a tárhelyre, majd meghívja a paraméterezett python scriptet.

Miután a python script lefutott, a kész képet visszatölti, illetve engedélyezi az extra paramétereket(ezeket bővebben a parancssoros részben fejtem ki).

### **Parancssoros rész:**

-----  
Paraméterek :

```
scan.py [-h] -i IMAGE -o OUTPUT [-m_o MEDIAN_O] [-g_o GAUSS_O]
        [-b_o BILINEAR_O] [-n_o INVERT_O] [-c_o CLAHE_O]
        [-kv_o KVANTALAS_O] [-m MEDIAN] [-g GAUSS] [-b BILINEAR]
        [-n INVERT] [-c CLAHE] [-kv KVANTALAS] [-t TERMINAL]
```

### **Magyarázat**

Megkövetelt paraméterek:

- i bemeneti kép elérési útja
- o kimeneti képek elérési útvonalja

Opcionális paraméterek:

A bemeneti kép manipulálását befolyásoló paraméterek (ezek a módosítások a képet a scannelés előtt módosítják)

- m\_o = Median filterezés-Só és bors zaj-ellen (az eredeti képen)
- g\_o = Gaussian zajcsökkentés-holmályosítja a képet
- b\_o = bilineáris filter
- n\_o = negatív kép
- c\_o = kontraszt normalizálás
- kv\_o = Kvantálás-quantization (alapértelmezett webes felületen 32)

A kimeneti képe(ke)t befolyásoló paraméterek (megj.: ha a transzformáció megghiúsult, ezek a lépések nem fognak lefutni)

- m = Median filterezés-Só és bors zaj-ellen
- g = Gaussian zajcsökkentés-holmályosítja a képet
- b = bilineáris filter
- n = negatív kép
- c = kontraszt normalizálás
- kv = Kvantálás-quantization (alapértelmezett webes felületen 16)

A paraméterek átvétele argparse-el történik:

```
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
                help="A scannelendő kép elérési útja")
```

Új paraméter bevezetésénél csak felveszünk egy új opciót az add\_argument funkcióval.

Személy szerint én a funkció orienált programozás híve vagyok.

Ebből következik hogy a script felépítése is erre a programozási módszerre épül. A paramétereknek külön funkció van definiálva, melyek átláthatóbbá teszik a kódot.

Egy új módszer beépítésénél elég, ha létre hozunk egy új függvénydefiníciót, hozzáadunk egy új paramétert a többi mellé és elhelyezzük a kódban.

### **Paraméterek működése:**

#### **Median filterezés:**

Itt a cv2.medianBlur () függvény az összes képpont középértékét veszi fel a kernel területen,

és a központi elem helyébe ez a középérték. Ez rendkívül hatékony a képek só és bors zajának ellen.

A középső elmosódásban a központi elemet mindig a kép néhány pixelértéke helyettesíti.

Ez hatékonyan csökkenti a zajt. A kernel mérete pozitív páratlan egész szám legyen.

#### **Gaussian filterezés:**

Ez a `cv2.GaussianBlur ()` függvény segítségével történik.

Meg kell adnunk a kernel szélességét és magasságát, melynek pozitívnak és páratlannak kell lennie.

A Gauss-elmosódás nagyon hatékonyan távolítja el a Gauss zajt a képből.

#### **Bilineáris filter:**

A `cv2.bilateralFilter ()` nagyon hatékony a zaj eltávolításában, miközben élesen tartja az éleket. A művelet azonban lassabb a többi szűrőhöz képest.

Ez a Gauss-szűrő egyedül a tér függvénye, azaz a közeli képpontokat a szűrés során figyelembe veszik.

Nem veszi figyelembe, hogy a pixelek szinte azonos intenzitással rendelkeznek.

Nem veszi figyelembe, hogy a pixel él-pixel, vagy sem. Szóval elmosódik az élek is, amit nem akarunk.

#### **Negatív filter:**

A képből kivonjuk a létező pixel értékét, maximális pixelből, azaz 255-ből.

pl.: egy fekete pixelből(0) fehért(255) így csinálunk :  $255 - 0 = 255$

#### **Kontraszt normalizálás:**

Angolul Contrast-limited adaptive histogram equalization (röviden : CLAHE).

Ezért a probléma megoldásához adaptív hisztogram-kiegyenlítést alkalmaznak.

Ebben az esetben a kép kis blokkokra van osztva, úgynevezett "tile"-okra (a tile mérete alapértelmezés szerint 8x8 az OpenCV-ben).

Ezután a blokkok mindegyike hisztogram, amit a szokásos módon kiegyenlítenek.

Tehát egy kis területen a hisztogram egy kis régióra korlátozódik (hacsak nincs zaj).

Ha zaj van, akkor az erősítésre kerül. Ennek elkerülése érdekében kontrasztkorlátozást alkalmaznak.

Ha bármelyik hisztogram a megadott kontraszt határérték felett van (alapértelmezés szerint 40 az OpenCV-ben),

akkor ezeket a képpontokat levágják és egyenletesen osztják el más tartályokra a hisztogramkiegyenlítés alkalmazása előtt.

A kiegyenlítés után a tile határaiban lévő tárgyak eltávolítása érdekében bilinéris interpolációt alkalmazunk.

#### **Kvantálás-quantization filter:**

A számítógépes grafikában a színes kvantálást vagy a színes kép kvantálást a színterek esetében kvantálják;

ez egy olyan folyamat, amely csökkenti a képben használt különböző színek számát, általában azzal a szándékkal,

hogy az új kép legyen olyan vizuálisan hasonló, mint az eredeti képhez.

Az 1970-es évektől kezdve tanulmányozták a bittérképeken a színes kvantálást végrehajtó számítógépes algoritmusokat.

A színes kvantálás kritikus fontosságú a sok színnel rendelkező képek megjelenítéséhez olyan eszközökön,

amelyek csak korlátozott számú színt jeleníthetnek meg, általában a memóriakorlátok miatt,

és lehetővé teszi bizonyos képek hatékony tömörítését.

## A tesztgépek hardware információi:

Intel Xeon X3440 @ 4x 2.5GHz  
8GB-ram

-----  
Intel Xeon X5670 @ 2x 2.933GHz  
4GB-ram

-----  
Intel Xeon E5620 @ 4x 2.4GHz  
8GB-ram

-----  
Intel Xeon E5-2630 v4 @ 4x 2.2GHz  
8GB-ram

-----  
AMD Ryzen 5 2600 Six-Core @ 12x 3.5G  
32GB-ram

-----  
Core™ i5 3320M  
8GB-ram

-----  
Core™ i5 2520M  
8GB-ram

-----  
Raspberry Pi 3 Model B  
64bit ARMv7 Quad Core Processor @1.2GHz  
1GB-ram

## Minimum követelmények:

1.0 GHz Intel Core 2 Duo (vagy AMD CPU), illetve hasonló teljesítményű armhf processzorok  
(ajánlott minél nagyobb single core teljesítményű processzor használata a sebesség javítása miatt).

1GB for XP / 2GB RAM for Vista/Win 7/Win10 / linuxon disztribúciótól függetlenül min 2GB ajánlott  
(A képek a memóriába lesznek betöltve szóval ez függ a képmérettől).

## ----- Követelmények, a képet illetően -----

### Az optimális Kép főbb jellemzői:

Az kép tartalmazza a teljes objektumot, melyet a felhasználó be szeretne scannelni.

A fényviszonyok egységesek.

A kép nem homályos.

A kép nem pixeles.

A képen az objektumot nem takarja el/ki semmi.

Az objektum nincs árnyék alatt.

Az objektum megfelelő szögben van befotózva.

Megfelelő formátumú a kép (jpeg, jpg, png...)

### A be scannelést, gátló tényezők:

Az objektum színárnyalatossal, kontúr nélküli.

Az objektum beleolvad a környezetébe.

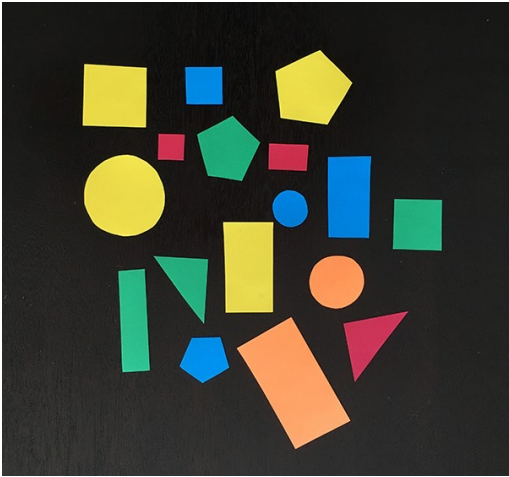
Nem jó formátumú kép (PL.: GIF).

Az alkalmazás olyan képek scannelésére alkalmas, ahol az objektum erős, jól elkülöníthető kontúrokkal rendelkezik.

Ezekre itt van egy pár péda:



->



->

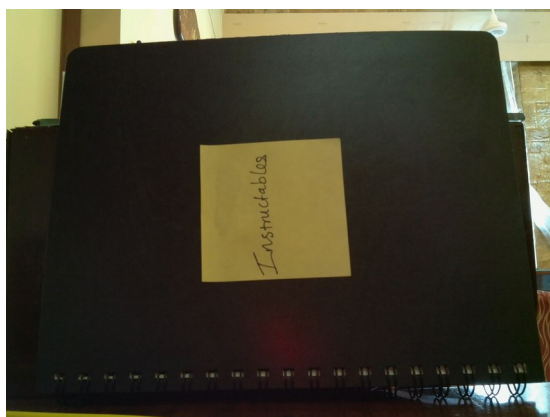


->

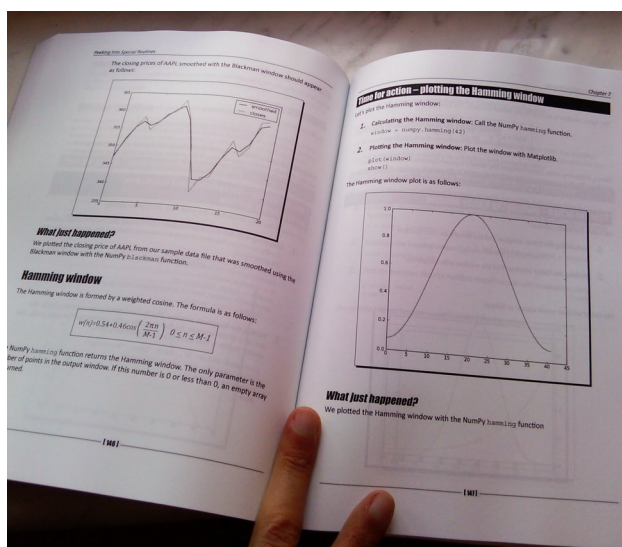
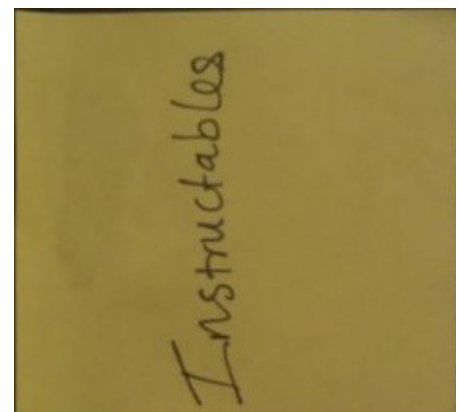




→



→



→

$$w(n) = 0.54 + 0.46 \cos\left(\frac{2\pi n}{M-1}\right) \quad 0 \leq n \leq M-1$$

-----Források-----  
[https://docs.opencv.org/3.1.0/d4/d13/tutorial\\_py\\_filtering.html](https://docs.opencv.org/3.1.0/d4/d13/tutorial_py_filtering.html)  
[https://docs.opencv.org/3.1.0/d5/daf/tutorial\\_py\\_histogram\\_equalization.html](https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html)  
<https://docs.python.org/3/howto/argparse.html>  
[https://people.csail.mit.edu/sparis/bf\\_course](https://people.csail.mit.edu/sparis/bf_course)  
<https://www.pyimagesearch.com>  
[https://www.youtube.com/watch?v=C\\_zFhWdM4ic](https://www.youtube.com/watch?v=C_zFhWdM4ic)  
<https://pinetools.com/add-noise-image>  
 illetve az órai példák