Képfeldolgozás a gyakorlatban

Képjavítás

Szeghalmy Szilvia

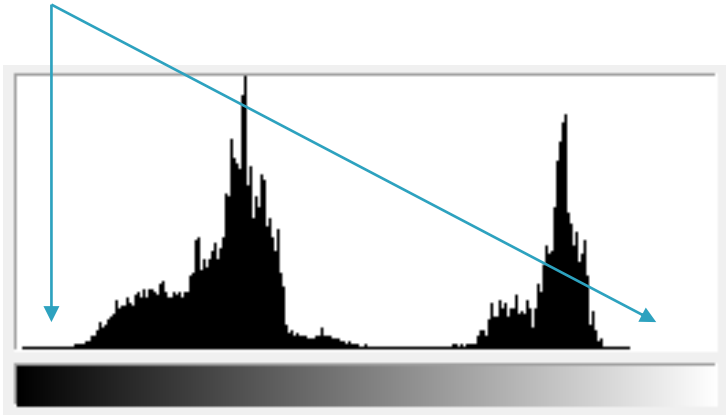
Tartalom

- Hisztogram transzformáció
  - kontraszt nyújtás
  - hisztogram kiegyenlítés
  - hisztogram kiegyenlítés színes képekre
- Zajcsökkentő eljárások
- Feladatok

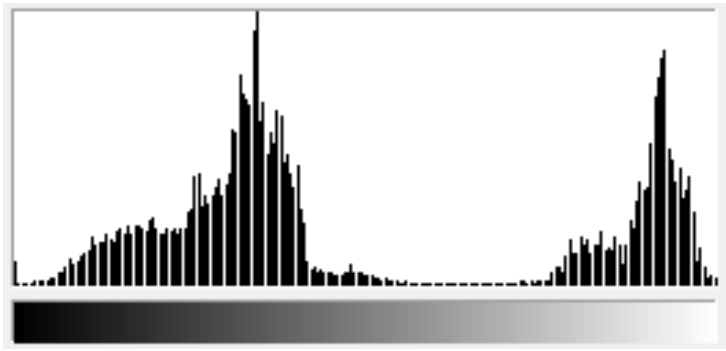
Kontraszt nyújtás

- A részletek láthatóbbá tétele:

kihasználatlan intenzitástartomány



hisztogram



hisztogram a kontrasztjavítás után

Kontraszt nyújtás

- Egy intenzitástartomány [ah; fh] leképezése a kép elméleti intenzitástartományára:

$$I' = (I - ah)\frac{255}{fh - ah}$$

  - Megvalósítás OpenCV-ben túlterhelt mátrixoperátorokkal.
    - Az OpenCV (C++ alatt) biztosítja a pontonkénti műveleteknél, hogy a tartományon belül maradjunk.
    - pl.: img.at<uchar>(0, 0) += 300; eredményeként 255-ös érték kerül a mátrixba.

- **Hisztogramnyújtás**:
  - A kontrasztnyújtás azon esete, amikor *ah* a képen ténylegesen előforduló legkisebb, *fh* a képen ténylegesen előforduló legnagyobb intenzitásérték.
  - Probléma: a kiugró értékek gyakran zajok => fontos a kép szűrése (később)

Hisztogram nyújtás

- Töltse le és olvassa be az alábbi képeket az e-learningből:

  debrecen_deep.png

- Az alábbi képletet használva végezzen hisztogramnyújtást

$$\text{dest} = (\text{img} - \text{ah})\frac{255}{\text{fh} - \text{ah}}$$

  - ah: a minimális intenzitásértéket
  - fh: a maximális intenzitásértéket

A szélsőértékek megkeresésének módja OpenCV-ben:

```
double min, max;
cv::minMaxLoc(img, &min, &max);
```

Kontraszt nyújtás

- Próbálja ki a programot az alábbi képre is:

  dark_img.jpg

- Jelenítse meg a *dark_img hisztogramját (képszerkesztőt is használhat)*
- Miért nem működött az eljárás?

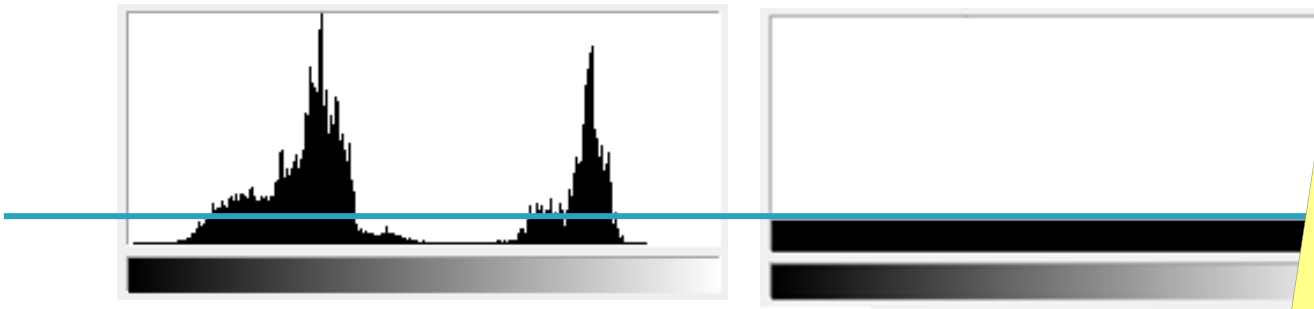
- Most a hisztogram képe alapján válassza meg az *fh* értékét. Az ah maradjon a korábbi.
- Végezze el OpenCV-ben a kontrasztnyújtást

$$\text{dest} = (\text{img} - \text{ah})\frac{255}{\text{fh} - \text{ah}}$$

  - fh: most a hisztogram képe alapján manuálisan választott érték

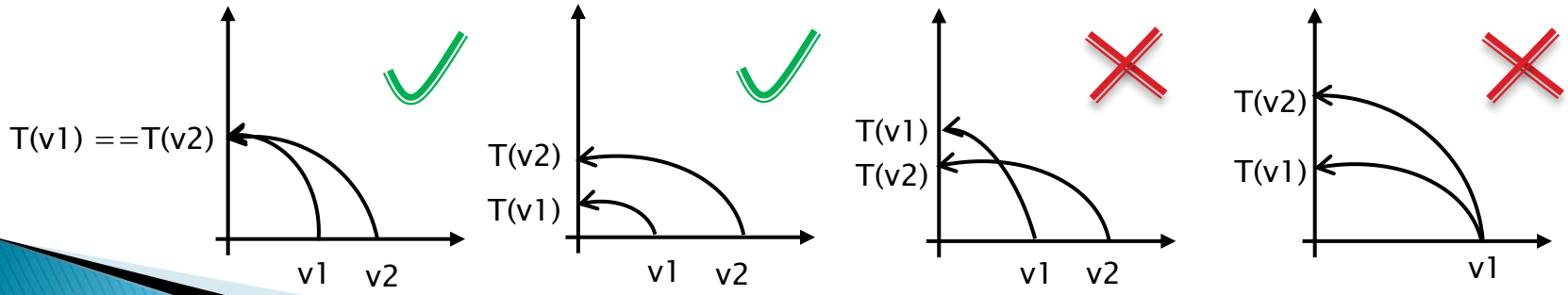
Hisztogram kiegyenlítés

- A cél ugyanaz: jobban kihasználni a rendelkezésre álló intenzitástartományt
- Próbáljuk egyenletesen eloszlatni az értékeket



Ilyen csak a mesében lesz az eredmény. Hiába van egy értékből túl sok, nem bontjuk majd ketté, hogy "beférjen a vonal alá".

  - ehhez egy T transzformációt használunk, mely
    - monoton nemcsökkenő (megőrzi az értékek sorrendjét): if v1 < v2 than T(v1) <= T(v2)
    - ha v1 = v2, akkor T(v1) = T(v2) (fordítva nem igaz)
    - megőrzi a tartományt (ha eredetileg [0, 255], akkor a T végrehajtása után is az)



Histogram kiegyenlítés

- A *T* transzformáció:

$$T(v) = \sum_{i=0}^{i \le v} h[i] \frac{L}{N}$$

  - L: az intenzitásértékek maximuma (szintek: {0, 1, ..., L})
  - N: a képpontok száma
  - h: hisztogram, ahol a *h[i] az i* intenzitású pixelek gyakorisága

- A transzformáció alkalmazása:

  dest(p) = T(src(p))

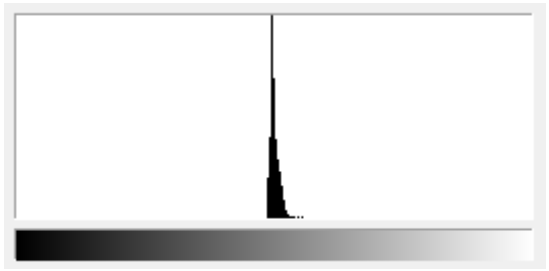
- OpenCV:

```
equalizeHist(InputArray img, OutputArray dest);
```

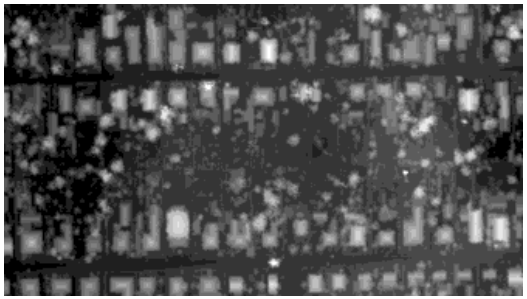
Feladat – hisztogram kiegyenlítés

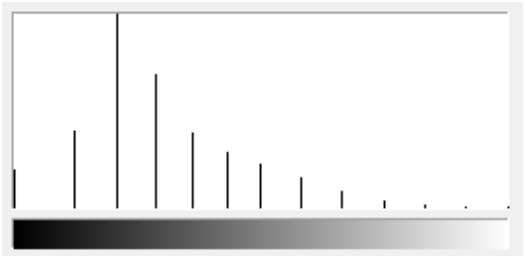
- Alkalmazzon hisztogramkiegyenlítést a debrecen_deep.png vagy a dark_img.jpg képre.
- Szürkeskálában olvassa be a képet! (vagy konvertája át)
- Hajtsa végre a transzformációt.

debrecen_deep



Hisztogram kiegyenlítés utáni





Feladat

- Hisztogram kiegyenlítés színes képekre

  https://en.wikipedia.org/wiki/File:Indian_hybrid_Orange.jpg

- Bontsuk csatornákra a képet

```
split( InputArray img, InputArrayofArrays dest);
```

  vector<Mat>

- Hajtsuk végre a transzformációt minden csatornára

  equalizeHist(InputArray src, OutputArray dest);

- Olvasszuk össze a csatornákat.

```
merge(InputArrayofArrays channels, OutputArray dest);
```

- Miért olyan szörnyű az eredmény? Hogyan változik a kép hisztogramja az egyes csatornák esetén?

Feladat: Hisztogram kiegyenlítés színes képekre

- Térjen át olyan színtérre, ahol elválik az színezet és a világosságinformáció.
  - Pl.: HSV, Lab, YCbCr, …
- Bontsa szét a képet csatornákra
- Végezze el a hisztogramkiegyenlítést a világosságinformációt tartalmazó csatornára (csak arra)
  - V (HSV esetében)
  - L (Lab esetében)
  - Y (YCbCr esetén)
- Olvassza össze a csatornákat
- Térjen vissza BGR-be.

Zaj

- Zaj: a képet terhelő elváltozás

$$img_{zajos} = img_{eredeti} + zaj$$

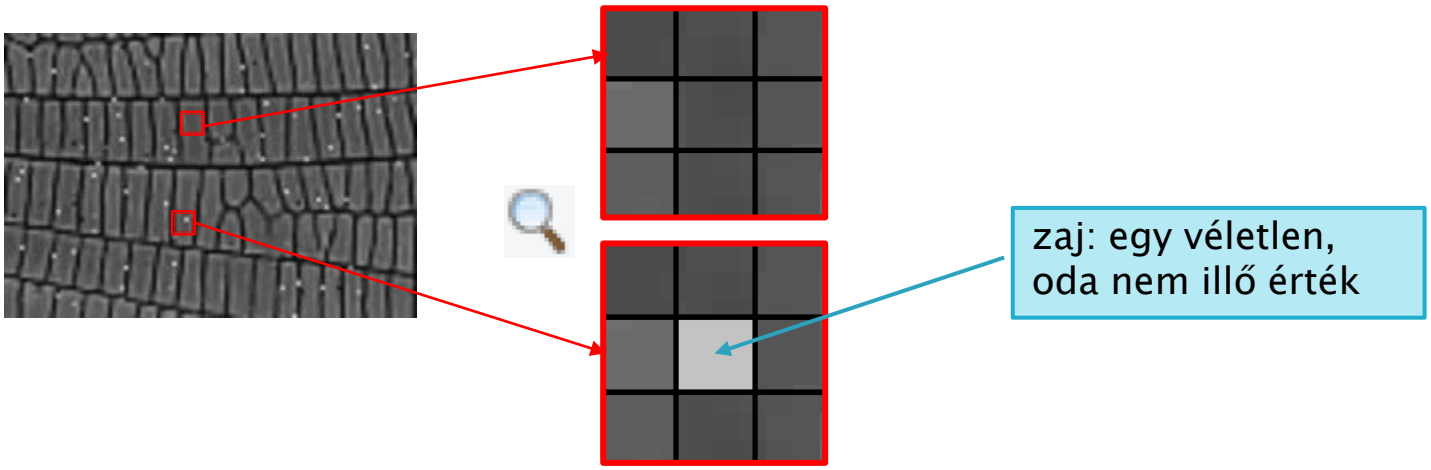
- Mikor keletkezhet?
  - a készítés során
  - az adat átvitel során
  - a tárolás során
  - a kép feldolgozása során

- A cél a zaj csökkentése, az értékes információk megőrzése mellett.

- Miért akarjuk csökkenteni?
  - esztétikai okokból
  - nehezítheti a képfeldolgozást
  - rontja a tömöríthetőséget

Zajszűrés

- A zajszűrés alapvető feltételezése: egy pont és a pont környezete hasonló.



  - Probléma: vékony vonalak, objektum szélek!

Zajszűrés

- Számoljuk ki a lokális környezet alapján, hogy milyen értéknek kellene az adott pontban állnia, és írjuk be a vizsgált pont értékeként.

- `blur(InputArray src, OutputArray dest, Size kernelSize);`
  - környezeti átlagolás: a képponttól távoli pontok is erősen hatnak

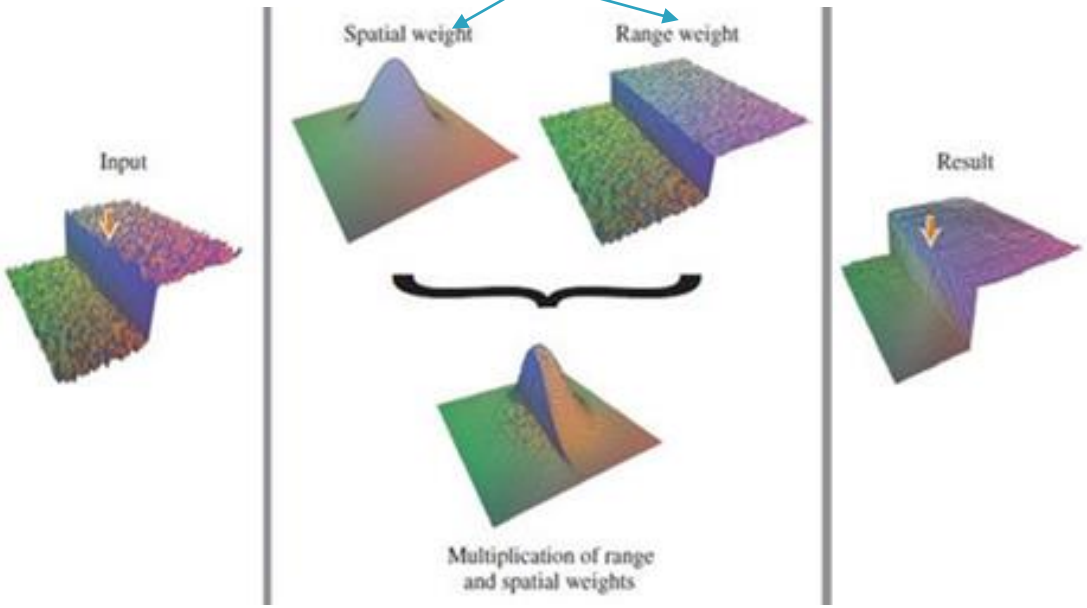
- `GaussianBlur(InputArray src, OutputArray dest, Size kernelSize, int sigmax, int sigmay);`
  - súlyozott átlag: a közeli pontokat jobban figyelembe vesszük, mint a távoliakat

- `medianBlur(InputArray src, OutputArray dest, int kernelSize);`
  - a környező pontok mediánjával írjuk felül (só/bors zajra, mélységképre)

Zajszűrés

```
bilateralFilter(InputArray src, OutputArray dest,
        int kernelSize, double sigmaColor, double sigmaSpace);
```

- Két kernel szorzataként áll elő:
  - Az egyik kernel felel azért, hogy a közeli pixelek erősebben számítsanak (*sigmaSpace*)
  - A másik azért, hogy azok a pixelek legyenek figyelembe véve, melyek színe hasonló az eredetihez (*sigmaColor*).
- A kernel az éppen vizsgált pont környezetétől függ:
  - Kevéssé mossa el az éleket
  - Lassú !!!
- Valós idejű alkalmazáshoz:
  - kernelSize = 5
  - erős elmosáshoz, akár 150-nél is nagyobb sigma



http://www.coldvision.io/2016/01/21/bilateral-filtering-with-cuda-and-opencv-3-x/

Feladat

- Töltse le az e-learningből az alábbi képeket:

  szita2.png, melyseg.png

- Készítsen egy ablakot "csúszkával":

```
int main(){

    string menu = "tool";
    cv::namedWindow(menu, WINDOW_NORMAL);
    cv::resizeWindow(menu, Size(500, 50));

    int radius = 1, sigma = 1;

    createTrackbar("radius", menu, &radius, 25);
    createTrackbar("sigma", menu, &sigma, 25);
```

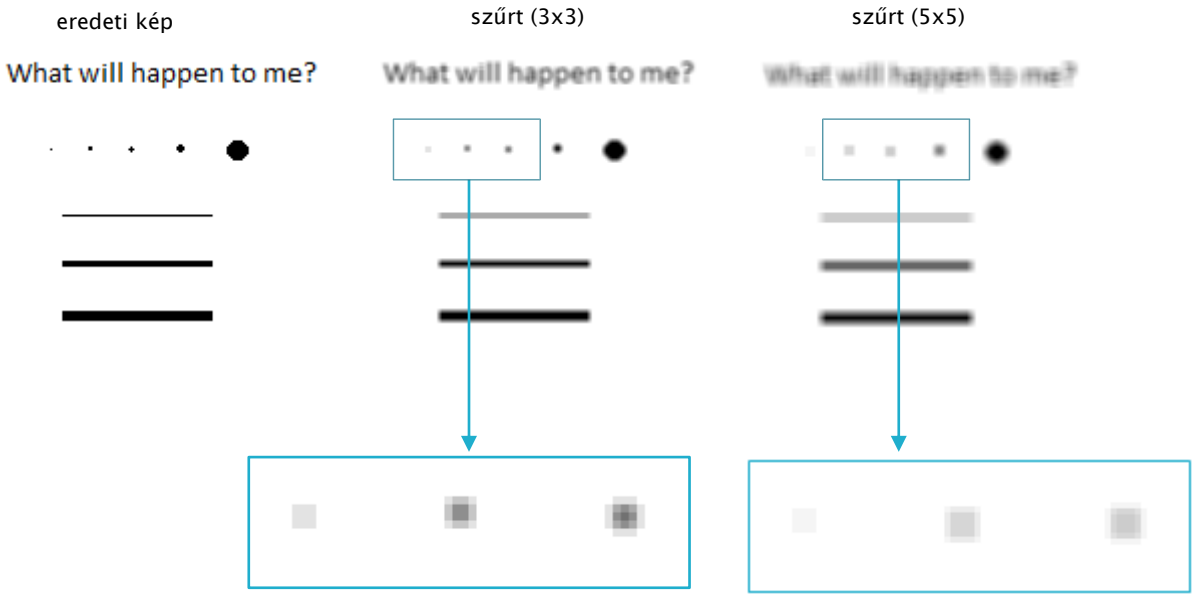
Feladat

- Töltse be valamely képet szürkeskálában.
- Jelenítse meg.
- Próbálja ki a szűrőket:

```
Mat blur_img, gauss_img, median_img, bilat_img;
while (cv::waitKey(20) != 'q') { //q-ra lép ki
   //mossa el a beolvasott képet a különböző szűrőkkel
   //a szűrő mérete legyen: 2*radius+3 vagy Size(2*radius+3, 2*radius+3)
   //a BilateralFilter-nél:
            d: 5
            sigmaColor: sigma*10  //csak, hogy látványos legyen
            sigmaSpatial: 2*radius+3
    //jelenitse meg a kepeket
}
```

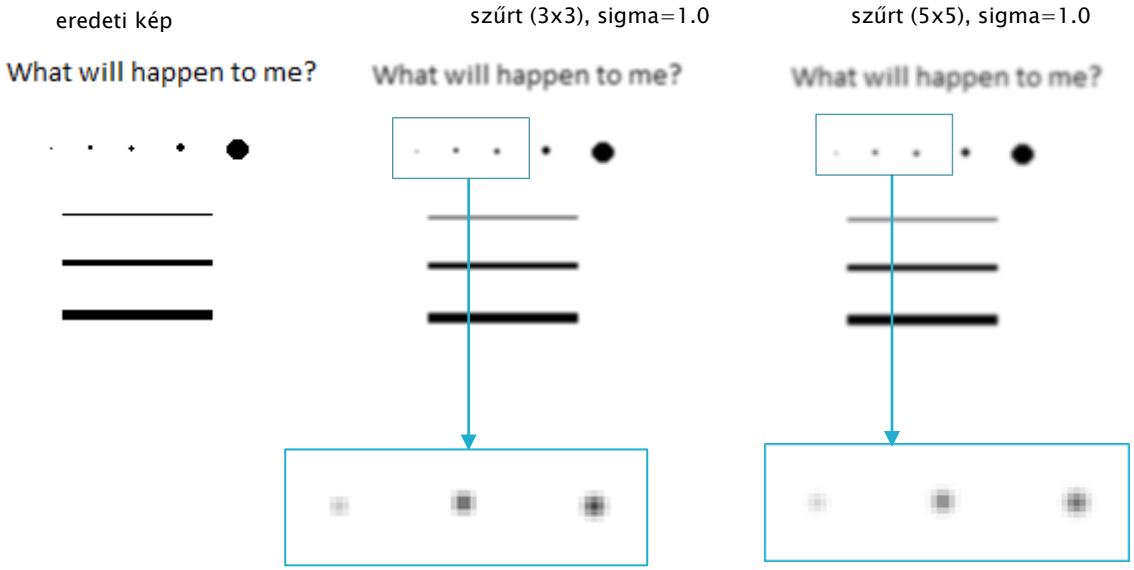
- Próbálja ki a másik képre is a programot. Melyik esetben, melyiket választaná?

A blur (lapos szűrő) hatása



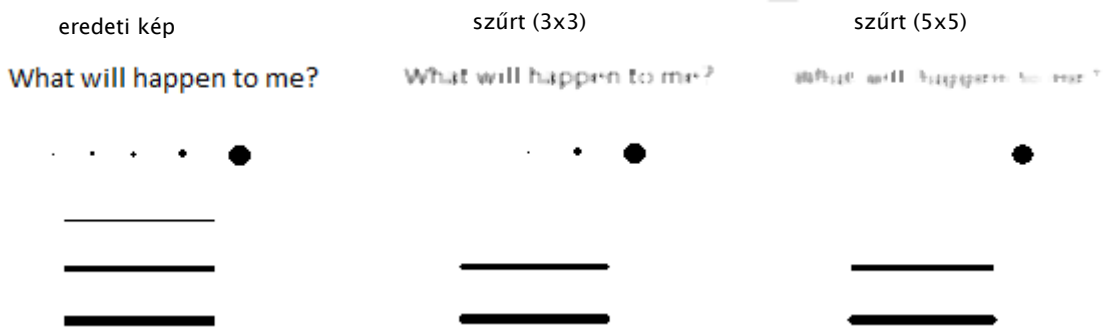
- **csökkenti** a zajt
- homályossá teszi a képet
- a képen eredetileg nem szereplő értékeket hozhat be. (Gondold át mi történik egy mélységképnél, fekete-fehér képnél.)

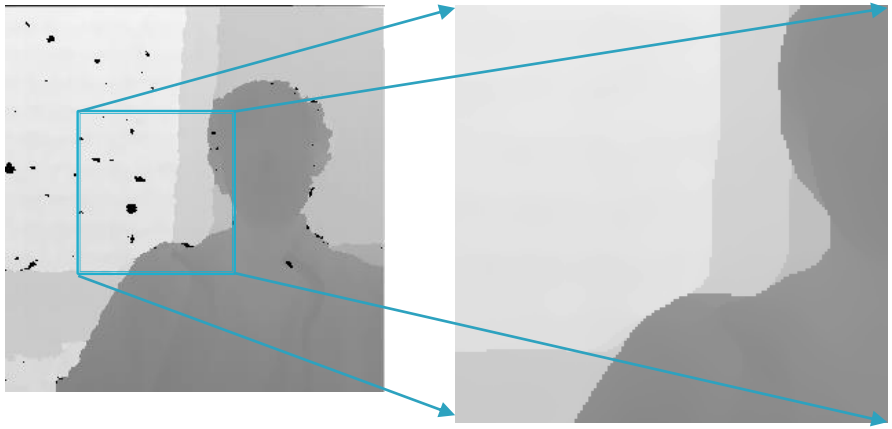
A Gauss szűrő hatása



- csökkenti a zajokat
- az elmosás hatással van az élekre, vonalakra, nagyobb szűrő, erősebb sigma homályossá teheti a képet.
- a képen eredetileg nem szereplő értékeket hozhat be. (Gondold át mi történik egy mélységképnél, fekete-fehér képnél.)

A medián szűrő hatása





- a megfelelő méretű szűrő teljesen **eltávolíthatja** a zajt
- erős hatása van az élekre, a kisebb objektumokat, vékony vonalakat is eltűntetheti a képről.
- nem hoz be új értékeket a képre, ezért a mélységképek, fekete-fehér képek esetén is használható.
- só-bors zaj esetén is ajánlott (3x3)

Gauss zaj

- Feltételezzük, hogy (független) normális eloszlású, 0 várható értékű, véletlen zaj rakódik a képre
- ha sok képünk van (I_1 , I_2 , ..., I_n) ugyanarról a jelenetről, akkor az előző szűrők alkalmazása helyett érdemes képpontonként átlagolni a képeket:

$$I(x,y) = \frac{1}{n}\sum_{i=1}^{n} I_i(x,y)$$

eredeti

normális eloszlású véletlen zaj 0 várható értékkel

zajos kép

Feladat: zajszűrés átlagoló szűrővel

- A "zaj" a Kossuth téren áthaladó embereket és járműveket jelenti.
- Töltse le a képsorozatot az e-learningből: KossuthSquare.zip
- Olvasson be egy képet
- Készítsen egy a képpel azonos méretű, fekete mátrixot (CV_64FC3):

```
Mat acc = Mat::zeros(...);
```

- Vegyen fel egy változót a képek számlálásához.
- Minden egyes képére: "SnapShot-20180731_173**715**.jpg"-től "..._173**918**"-ig
  - olvassa be a képet
  - ha a kép létezik:
    - növelje a képszámláló értékét
    - jelenítse meg és várakoztasson valamennyit
    - adja hozzá a képet az *acc* matrixhoz a következő függvénnyel:

```
accumulate( InputArray img, InputOutputArray acc);
```

- Konvertálja az *acc* Mátrixot CV_8UC3 képpé:

```
Mat dest;
acc.convertTo(dest, CV_8U, 1.0 / numFrames);
```


99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.
127 little bugs in the code...