# VEŽBA 5 – Potiskivanje šuma u slici

## Potrebno predznanje

- Poznavanje programskog jezika C
- 2D signali
- RGB i YUV prostori boja i konverzija između ova dva prostora
- 2D kovolucija
- Filtriranje slike

## Šta će biti naučeno tokom izrade vežbe

Tokom izrade ove vežbe upoznaćete se sa različitim vrstama smetnji koje se javljaju u signalima slike i videa. Naučićete na koji način je moguće primeniti filtriranje slike u prostornom domenu kako bi se izvršilo otklanjanje tih smetnji. Upoznaćete se sa primerom nelinearnog algoritma za otklanjanje šuma koji se primenjuje u slučajevima kada linearni sistemi obrade ne daju dovoljno dobre rezultate (npr kod impulsnog šuma)

## Motivacija

Uklanjanje smetnji i šuma iz signala slike i videa predstavlja veliku podoblast digitalne obrade signala. Tehnike za modelovanje šuma i algoritmi za otklanjanje istog imaju značajnu ulogu u sistemima potrošačke elektronike kada se teži prikazivanju slike vrhunskog kvaliteta ili u obradi rezultata različitih skenera i senzora kod kojih ponovljeno snimanje podrazumeva veliku cenu ili je čak neizvodljivo (u medicini, vojnoj industriji, svemirskim misijama). Još jedna od mogućih primena ove oblasti jeste i prilikom reparacije starih umetničkih dela, kada je potrebno ukloniti oštećenja nastala vremenom. Osnovni zadatak tehnika za otklanjanje šuma jeste uklanjanje nepoželjnog dela signala slike ili videa uz očuvanje ostalih detalja. U okviru ove vežbe biće obrađeni neki od šumova koji se javljaju u okviru signala slike i videa i neki odnačina za njihovo otklanjanje.

# Teorijske osnove

## Šum u slici

Šum u opštem slučaju predstavlja neželjenu stohastičku varijaciju signala. U okviru signala slike pod šumom smatramo razliku vrednosti pojedinih piksela u odnosu na njihovu pravu vrednost. Šum može nastati prilikom snimanja sadržaja, obrade ili prenosa signala. Različiti faktori mogu da izazovu pojavu šuma u slici. Neki od primera su:

- Uticaj faktora okruženja na senzor korišćen za snimanje signala slike
- Pregrejavanje ili fizička oštećenja senzora
- Smetnje u kanalu za prenos signala

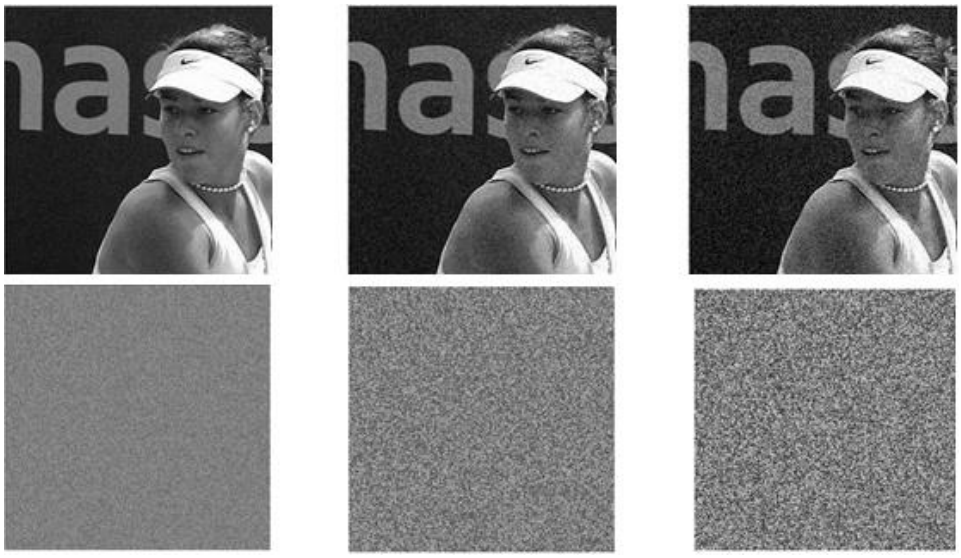
Broj oštećenih piksela i njihova razlika u odnosu na stvarnu vrednost određuju količinu prisutnosti šuma unutar slike.

Prema načinu na koji se kombinuju sa signalom tipove šuma možemo podeliti na dve grupe:

- Aditivni šum
- Multiplikativni šum

Neki od tipova šuma koji se često susreću unutar signala slike su:

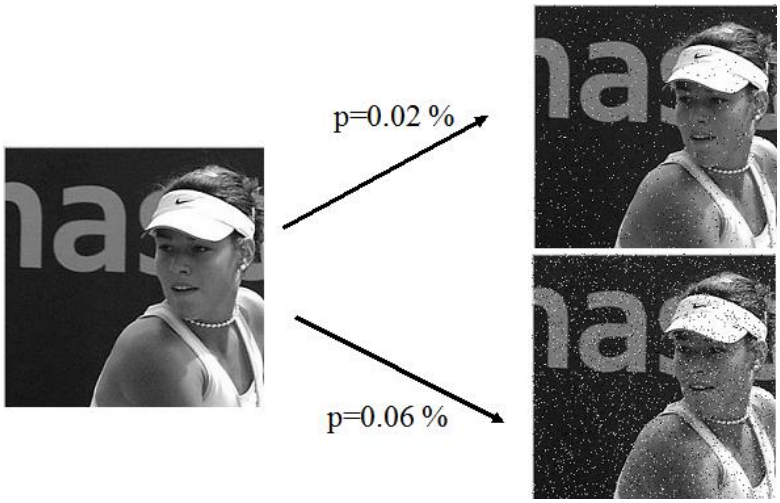
1. *Gausov šum* – predstavlja slučajan aditivni šum sa Gausovom raspodelom. Vrednost svakog piksela u zašumljenoj slici predstavlja sumu stvarne vrednosti piksela i slučajne vrednosti distribuirane sa Gausovom raspodelom. Intenzitet šume nezavisan je u odnosu na stvarne vrednosti piksela u bilo kojoj tački. Ovakva vrsta šuma se obično javlja na analognim komponentama kao što su senzori ili pojačavači signala.



Slika1 - SlikazašumljenaGausovimšumomiprikazsamogšuma u prostornomdomenuzavredonsti σ = 10, 20 i 30

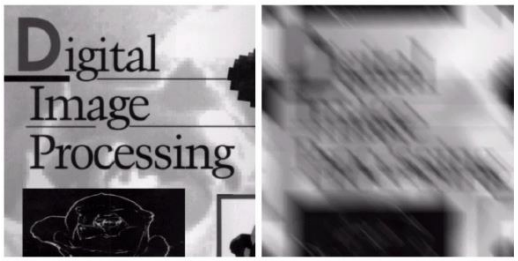
2. *Impulsni šum* – za razliku od prethodno pomenutog u slučaju impulsnog šuma nisu svi pikseli u slici zašumljeni već samo pojedini. Vrednost onih piksela koji su zašumljeni prmenjena je u maksimalnu ili minimalnu vrednost (u slučaju osmobitne monohromatske slike 0 ili 255). Ovakav šum manifestuje se kao pojava belih ili crnih tačaka u slici. Jedan od načina na koji može biti izazvan jeste oštećenjem senzora ili prisutnošću prašine na sočivu kamere. Intenzitet šuma može se modelovati kao:

$$d(v,h) = \begin{cases} 1 & verovatnoća\ p \\ 0 & verovatnoća\ 1-p \end{cases}$$




Slika2 - Impuslnišumzavrednosti p=0.02% i p=0.06%

3. *Zamućenjepokreta unutar slike* - nastaje kao posledica pomeraja kamere ili objekta prilikom snimanja slike. Manifestuje se pojavom da se na stvarnu vrednost piksela dodaje vrednost okolnih piksela u zavisnosti od pravca kretanja objekta.




Slika3 - Zamućenjepokretaunutarslike

*4.* *Blocking i Ringing efekat* – ova vrsta šuma nastaje kao posledica kompresije slike sa gubicima. *Blocking*efekat predstavlja posledicu blokovske obrade, tačnije činjenice da se kvantizacija elemenata vrši nad svakim blokom posebno. Ovo rezultuje greškom u slici na ivicama blokova obrade. Efekat zvonjenja (*ringing*) se dobija kvantizacijom ili odbacivanjem koeficijnata koji se tiču visokih frekvencija u okviru DCT ili vejvlet transformacije slike. U prostornom domenu ova pojava se manifestuje kao talasanje ili oscilacije oko oštrih ivica u ili kontura u slici.

Slika4 - Originalnaslika, slikasa blocking islikasa ringing efektom

## Algoritmi zauklanjanje šuma u slici

Uklanjanje špuma predstavlja važan zadatak u okviru obrade i analize signala slike. Postoji velik broj različitih algoritama za uklanjanje šuma. Koja metoda se koristi zavisi od tipa šuma koji je potrebno ukloniti. Mera kvaliteta algoritma za uklanjanje šuma jeste mogućnost tog tog algoritma da ukloni kompletan neželjeni signal iz slike uz očuvanje detalja unutar originalne slike. Bitna mera kod algoritama koji se primenjuju na obradu slike u realnom vremenu jeste i vreme potrebno da se algoritam izvrši. Postoje linearni i nelinerarni algoritmi.

### Linearni algoritmi

Primer prostog linearnog algoritma jeste filtriranje upotrebom **filtra usrednjivača**. Ovaj algoritam zasniva se na činjenici da se većina korisnog signala unutar slike nalazi u niskom delu spektra. Iz tog razloga filtriranje upotrebom nisko propusnog filtra eliminiše signal greške koji se nalazi u visokom delu spektra. Nedostatak ovog pristupa jeste to što se gube i detalji slike koji se nalaze u visokom delu spektra i dolazi do zamućenja slike.

Primer kernela filtra usrednjivača:

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Na slici 5 dat je primer sukcesivnog filtriranja slike zašumljene gausovim šumom upotrebom prikazanog kernela.

Slika5 - SukcesivnofiltriranjeslikezašumljeneGausovimšumomupotrebomfiltrausrednjivača

Drugi NF filter koji se često koristi za potiskivanje šuma je **Gausov filter**, nazvan po tome što su koeficijenti filtra definisani preko gausove funkcije:

$$h(n,k) = C \cdot e^{-\frac{(n-N/2)^2+(k-N/2)^2}{2\sigma^2}},$$

Pri čemu je konstanta C određena tako da filter ima jedinično DC pojačanje:

$$\sum_{n=0}^{N-1}\sum_{k=0}^{N-1} h(n,k) = 1.$$

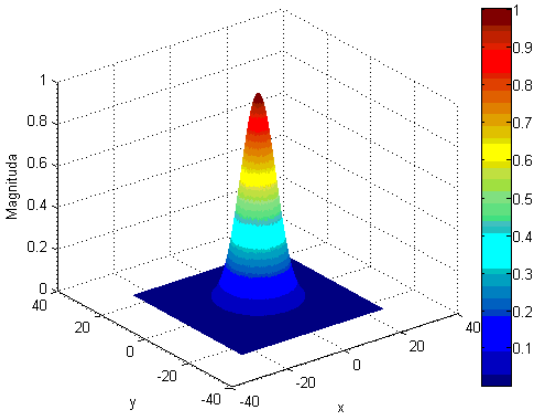
Prenosna karakteristika ovog filtra takođe je data Gausovom funkcijom i nema sporo-opadajuće lobove na visokim učestanostima.

Primer kernela Gausovog filtra veličine 5x5 za σ=10 dat je na slici 6.

σ=10

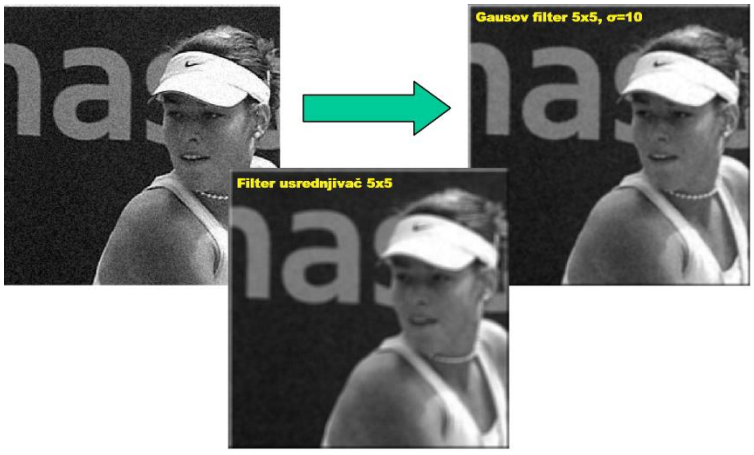
$\frac{1}{273}$

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 7 | 4 | 1 |
| 4 | 16 | 26 | 16 | 4 |
| 7 | 26 | 41 | 26 | 7 |
| 4 | 16 | 26 | 16 | 4 |
| 1 | 4 | 7 | 4 | 1 |




**Slika6 - Gausovfiltarzaσ=10**

Rezultati filtrarnja slike upotrebom filtra usrednjivača i Gausovog filtra istog reda dati su na slici 7.




**Slika7 - Primer filtriranjaslikeupotrebomGausovogfiltra**

Filtriranje slike se vrši primenom dvodimenzione diskretne konvolucije, kao što je to prethodno prikazano u vežbi 7. Koeficijenti filtera predstavljaju matricu zadatih dimenzije NxN gde je N neparan broj. Računanje koeficijenta Gausovog filtera, upotrebom programskog jezika C može se izvršiti implementacijom ranije predstavljene jednačine. Za računanje vrednosti $e^x$ može se iskoristiti funkcija:

- `double exp (double x);`

koja je deo standradne matematičke biblioteke, i čija se deklaracija nalazi u zaglavlju <*math.h*>

Koeficijent C iz pomenute jednačine jednak je recipročnoj vrednosti zbira svih koeficijenata. Njegova vrednost se može izračunati tako što se saberu svi koeficijenti i potom dobijenim broje podeli broj 1. Nakon toga, potrebno je još jednom proći kroz matricu sa koeficijentima i svaki od koeficijenata pomnožiti sa tom vrednošću.

```
void calculateGaussKernel(double kernel[], int N, double sigma)
{

    double C = 0;
for(int n = 0; n < N; n++)
    {
for(int k = 0; k < N; k++)
        {
            kernel[n*N+k] = exp( -((n - N/2)*(n - N/2) + (k - N/2)*(k - N/2))
                           /  (2 * sigma * sigma));
            C += kernel[n*N+k];
        }
    }

    C = 1.0 / C;

for(int n = 0; n < N; n++)
    {
for(int k = 0; k < N; k++)
        {
            kernel[n*N+k] *= C;
        }
    }
```

## Nelinearni algoritmi

Pomenuti pristup filtriranja slike upotrebom niskopropusnih filtara pokazao se kao neefikasan u slučaju potiskivanja impulsnog šuma. Pomenuti filtri uspevaju samo da umanje intenzitet impulsnog šuma ali ne i da ga u potpunosti otklone. Značajno efikasnije potiskivanje impulsnog šuma se postiže jednom nelinearnom obradom nazvanom **Median filter**. Ideja se bazira na činjenici da svaki pojedinačni impuls šuma značajno odstupa od okolnih vrednosti u originalnoj slici, one su uvek više grupisane uz sporiju promenu. Median filter je obrada koja se sastoji iz tri koraka za svaku tačku obrađene slike y(v,h).

1. U prvom koraku se izdvoji blok vrednosti susednih tačaka veličine NxN (N je neparno):

$$b_{v,h}(n,k) = x\left(v-(N-1)/2+n, h-(N-1)/2+k\right)\begin{matrix} n=0,\ldots,N-1 \\ k=0,\ldots,N-1 \end{matrix}.$$

2. Zatim se taj blok sortira po rastućim vrednostima u niz od $N^2$ vrednosti:

$$s_{v,h}(m) = sort\left(b_{v,h}\right) m = 0,\ldots,N^2-1.$$

3. Konačno se uzme vrednost središnjeg elementa toga niza za izlaznu vrednost:

$$y(v,h) = s_{v,h}\left(\frac{N^2-1}{2}\right).$$

Na slici 8 dat je primer filtriranja slike zašumljene impulsnim šumom upotrebom Median algoritma

**Slika8 - Primer uklanjanjaimpulsnogšumaupotrebom Median algoritma**

Sortiranje niza moguće je izvršiti na različite načine. Jedan od mogućih načina jeste sortiranje niza koristeći *bubble sort* algoritam. Realizacija ovog algoritma u programskom jeziku C data je sa:

```
int i, j;
bool swapped;
for (i = 0; i < n-1; i++)
{
swapped = false;
for (j = 0; j < n-i-1; j++)
 {
if (arr[j] > arr[j+1])
{
int tmp = arr[j];
     arr[j] = arr[j+1];
                arr[j+1] = tmp;
swapped = true;
}
   }

if (swapped == false)
break;
    }
```

S obzirom da je cilj sortiranja određivanje median vrednosti, nije potrebno algoritam sortiranja izvršavati do kraja. Sortiranje se može prekinuti onog momenta kada se odredi vrednost središnjeg elementa, odnosno elementa sa indeksom (n/2+1).

U slučajevima kada je brzina računanja median vrednosti značajna i veličina bloka za računanje median vrednosti konstantna, moguće je primeniti neku od metoda za brzo računanje median vrednosti. Jedna takva metoda, specijalizovana za računanje median vrednosti za veličinu niza N=3x3, data je na sledećem primeru:

```
PIX_SORT(a,b) { if ((a)>(b)) PIX_SWAP((a),(b)); }
pixelvalue opt_med9(int p[])
{
    PIX_SORT(p[1], p[2]) ; PIX_SORT(p[4], p[5]) ; PIX_SORT(p[7], p[8]) ;
    PIX_SORT(p[0], p[1]) ; PIX_SORT(p[3], p[4]) ; PIX_SORT(p[6], p[7]) ;
    PIX_SORT(p[1], p[2]) ; PIX_SORT(p[4], p[5]) ; PIX_SORT(p[7], p[8]) ;
    PIX_SORT(p[0], p[3]) ; PIX_SORT(p[5], p[8]) ; PIX_SORT(p[4], p[7]) ;
    PIX_SORT(p[3], p[6]) ; PIX_SORT(p[1], p[4]) ; PIX_SORT(p[2], p[5]) ;
    PIX_SORT(p[4], p[7]) ; PIX_SORT(p[4], p[2]) ; PIX_SORT(p[6], p[4]) ;
    PIX_SORT(p[4], p[2]) ; return(p[4]) ;
}
```

# Zadaci

## Zadatak 1:

Realizovati funkciju:

- `voidperformMovingAverage(uchar input[], int xSize, int ySize)`

Funkcija vršifiltriranje upotrebom filtra usrednjivača upotrebom kernela dimenzija 5x5. Proveriti rezultat filtriranja slika sa prisutnim Gausovim šumom i slika sa impulsnim šumom. Za filtriranje koristiti funkciju *convolve2d* realizovanu u okviru vežbe 7.

## Zadatak 2:

Realizovati funkciju:

- `void performGaussFilter (uchar input[], int xSize, int ySize, double sigma)`

Funkcija vrši filtriranje upotrebom Gausovog filtra dimenzija 5x5. Parametar sigma predstavlja varijansu raspodele. Kao parametar sigma potrebno je proslediti vrednost grafičke kontrole *params[0]*.

Za izračunavanje kernela Gausovog filra realizovati funkciju:

- `void calculateGaussKernel(double kernel[], int N, double sigma)`

Za računanje koristi jednačinu datu u okviru teorijskih osnova. Potrebno je izračunati vrednosti svih koeficijenata pri čemu se zanemari koeficijent C u jednačini. Potom je potrebno izvršiti sumu svih koeficijenata i svaki koeficijent podeliti sa tom sumom kako bi se dobilo jedinično pojačanje.

Proveriti rezultat filtriranja slika sa prisutnim Gausovim šumom i slika sa impulsnim šumom za različite vrednosti sigma.

## Zadatak 3:

Realizovati funkciju:

- `void performMedianFilter (uchar input[], int xSize, int ySize, intN)`

Funkcija vrši obradu slike upotrebom Median algoritma nad blokovima 3x3 (parametar N zanemariti). Voditi računa da je neophodno izvršiti proširenje slike pre blokovske obrade. Proveriti rezultat filtriranja slika sa prisutnim Gausovim šumom i slika sa impulsnim šumom.

## Zadatak 4:

Modifikovati rešenje iz zadatka 3 tako da funkcija koristi parametar N koji predstavlja dimenziju bloka čija se median vrednost računa. Proveriti rezultat obrade za vrednosti N= 3, 5 i 7.