

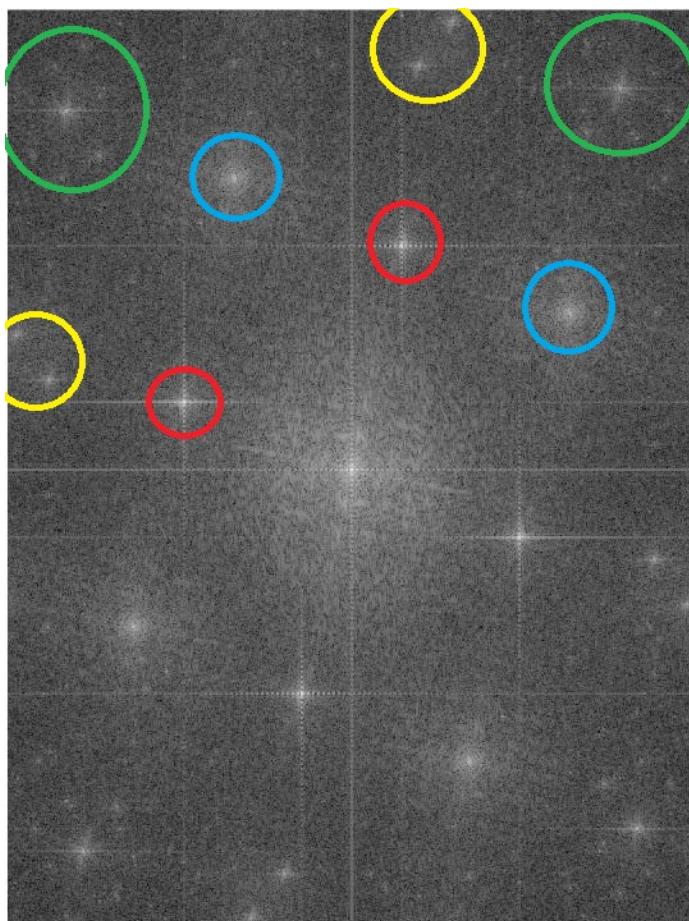
Univerzitet u Beogradu Elektrotehnički fakultet

Digitalna obrada slike

Drugi domaći zadatak

Poboljšanje kvaliteta štampane slike

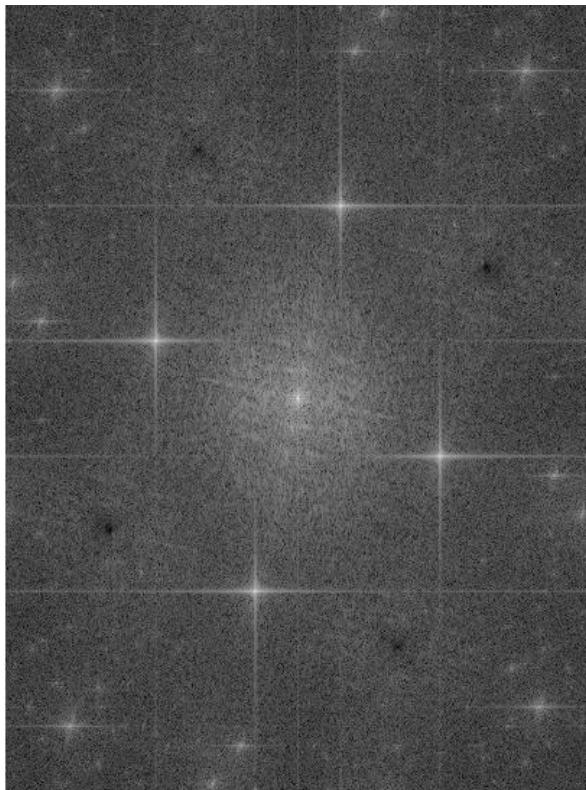
Za filtriranje slike su korišćene različite kombinacije selektivnog i low pass filtriranja. Selektivno filtriranje odlično uklanja tačkice, ali slika i dalje ima dosta šuma na višim učestanostima koji nije pravilan i ne može efikasno da se ukloni na ovaj način. Sa druge strane, samo low pass filtriranje odlično uklanja šum, ali daje jako mutnu sliku. Zato je izabrana kombinacija ova dva – šum koji je pravilan i na relativno nižim učestanostima je selektivno filtriran, a nepravnost na višim učestanostima su izbačene korišćenjem Gausovog NF filtra. Na slici 1.1 je prikazan spektar originalne slike, na slikama 1.2, 1.3, 1.4, 1.5 rezultati selektivnog filtriranja, a na slici 1.6 rezultati niskofrekventnog filtriranja.



Slika 1.1 Spektar originalne slike

Crvenom i plavom bojom su obeleženi delovi spektra koji najviše kvare kvalitet slike. Dakle, bilo kakvo smisleno filtriranje mora da izbacuje ove komponente. Ovo je razlog što isključivo NF filter ne dolazi u obzir – njegova granična učestanost bi onda bila premala i do dovodi do jako mutne slike.

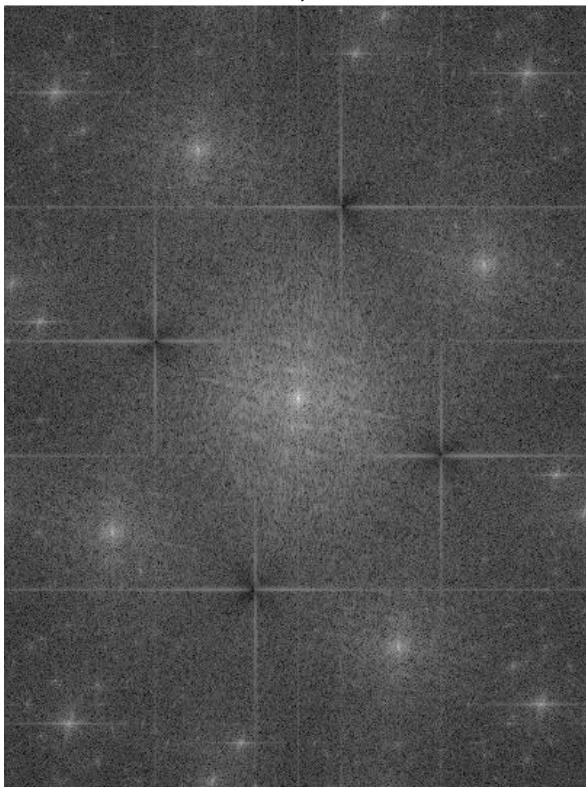
Takođe, ako se pogleda spektar na višim učestanostima, lako se uočava da šum obeležen zelenom i žutom bojom nema iste pravilnosti kao prethodna dva i da je smislenije ukloniti ove komponente niskofrekventnim filtriranjem.



Slika 1.2.1. Spektar slike nakon primene "cnotch btw" filtra



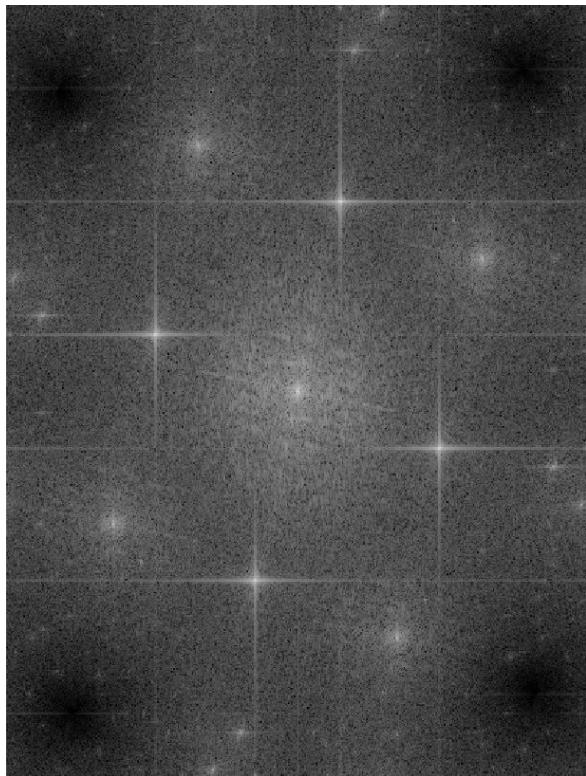
Slika 1.2.2. Slika nakon primene "cnotch btw" filtra



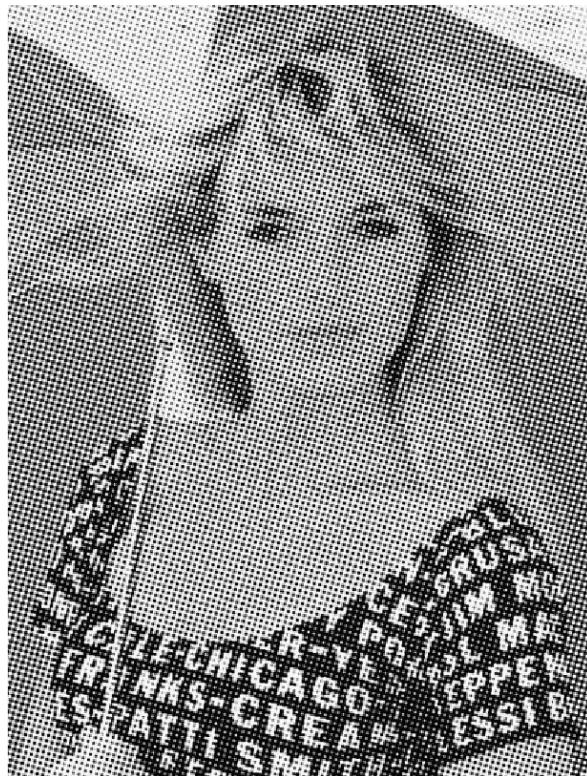
Slika 1.3.1. Spektar slike nakon primene "cnotch gaussian" filtra



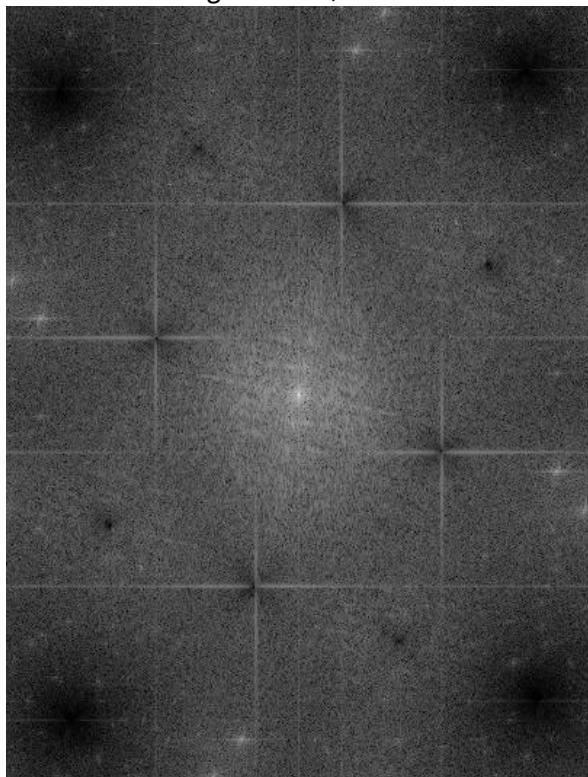
Slika 1.3.2. Slika nakon primene "cnotch gaussian" filtra



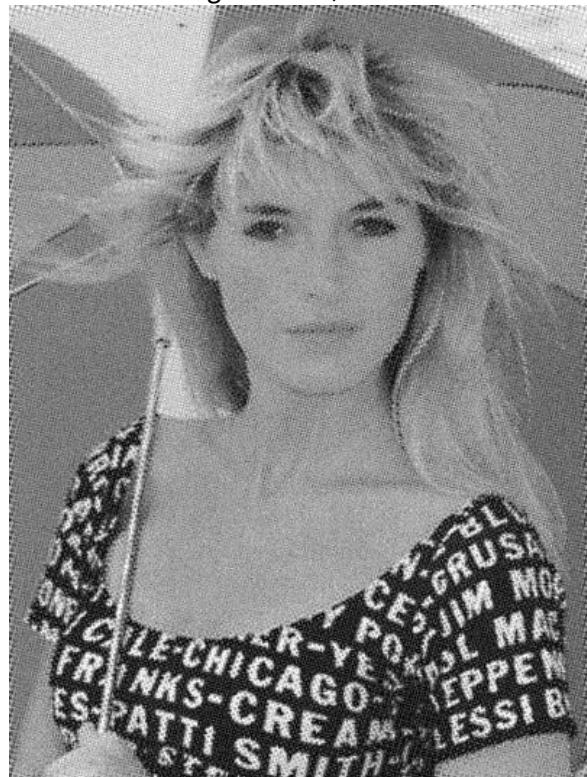
Slika 1.4.1. Spektar slike nakon primene "cnotch gaussian" filtra



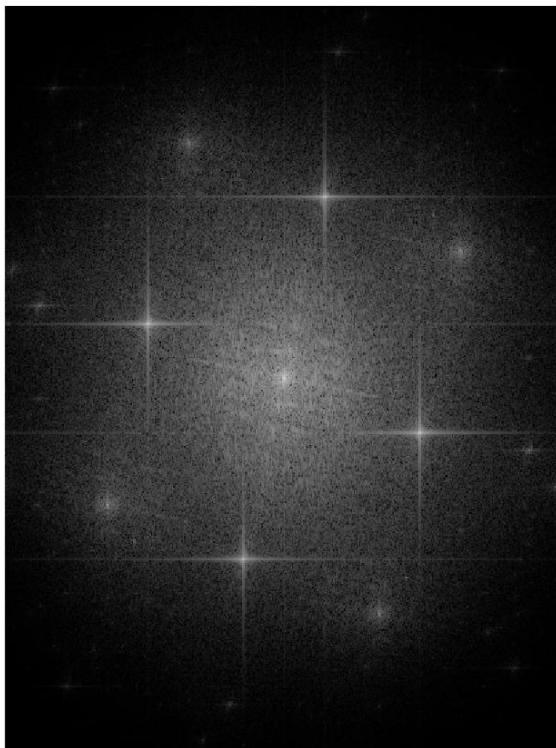
Slika 1.4.2. Slika nakon primene "cnotch gaussian" filtra



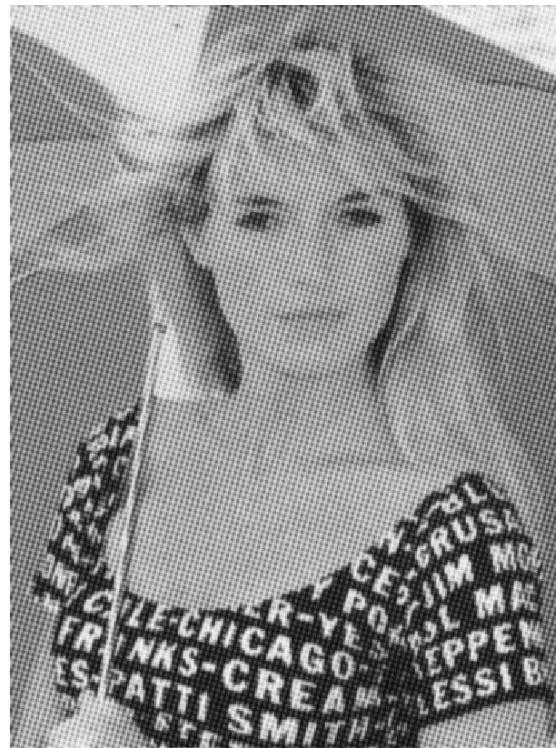
Slika 1.5.1. Spektar nakon primene sva tri filtra



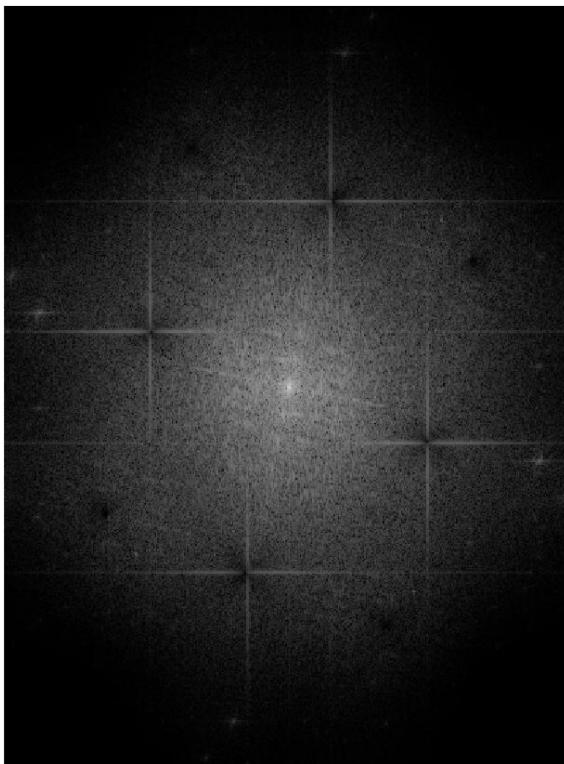
Slika 1.5.2. Slika nakon primene sva tri filtra



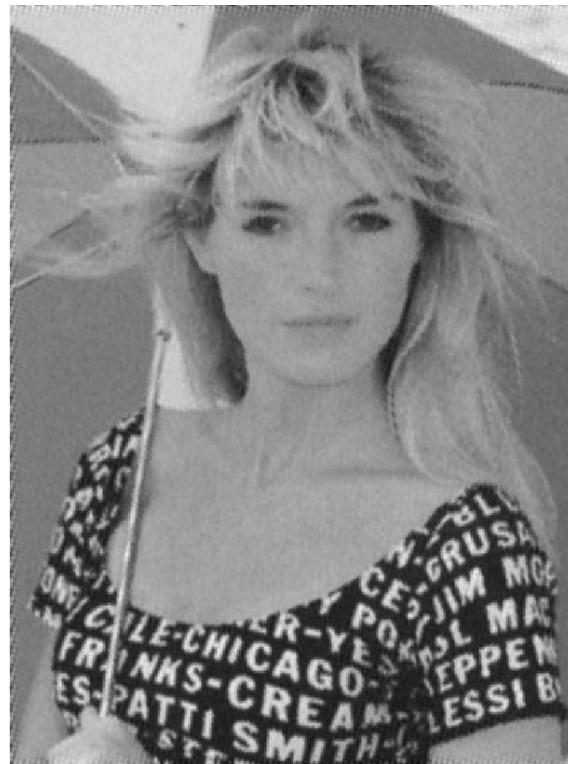
Slika 1.6.1. Spektar nakon primene Gausovog NF filtra



Slika 1.6.2. Slika nakon primene Gausovog NF filtra



Slika 1.7.1. Spektar nakon primene selektivnog i NF filtriranja



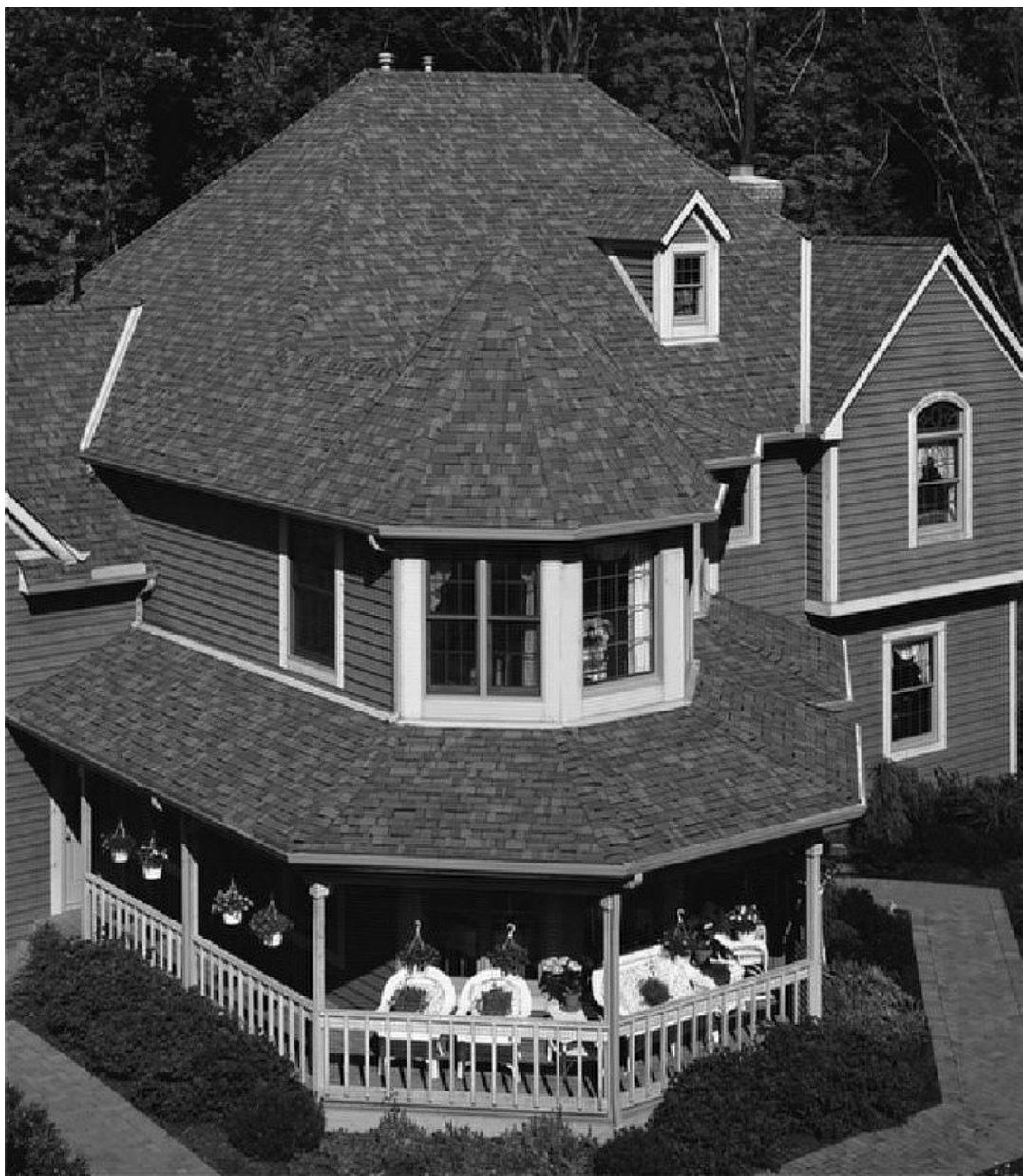
Slika 1.7.2. Slika nakon primene selektivnog i NF filtriranja

Kada se primenjuje NF filter, zbog periodičnosti slike, ivice slike kada se slika proširuje nulama su gotovo iste kao kada se direktno primeni filter. Na slici 1.5 je prikazan pokušaj poboljšanja slike samo uz pomoć selektivnog filtriranja. Kao što se vidi, uklanjanje ovih komponenti nije bilo dovoljno da se dobije slika razumnog kvaliteta, pa je bilo logičnije primeniti NF filter. Naravno, filter prikazan na slici 1.4. je postao suvišan onda kada se uveo low pass.

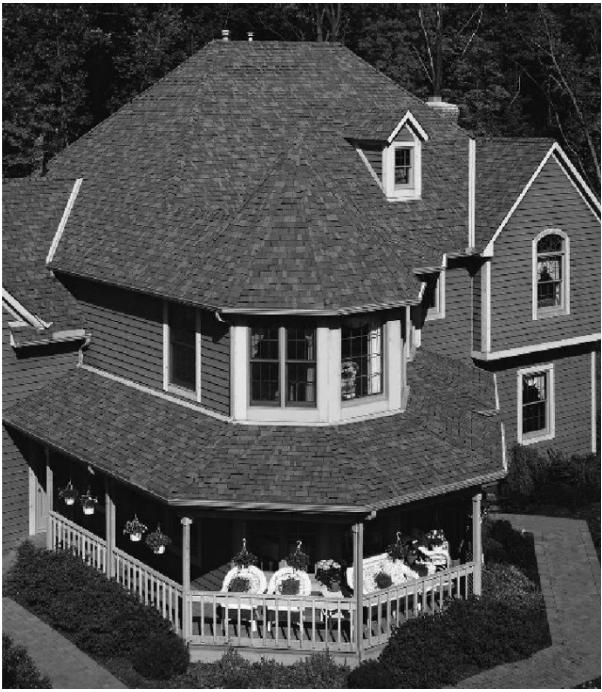
Funkcija koja vrši decimaciju slike

Funkcija dos_downscale(*I,s*) kao parametre uzima sliku i ceo broj s govori koliko puta treba smanjiti sliku. Pre smanjivanja slika se dopunjava nulama a zatim filtrira idealnim NF filtrom da bi se izbeglo preklapanje u spektru. Granična učestanost je izabrana tako da bude najveća moguća, a da ne dođe do preklapanja u spektru, dakle prema formuli $f_{\max} = \min(M/s, N/s)$.

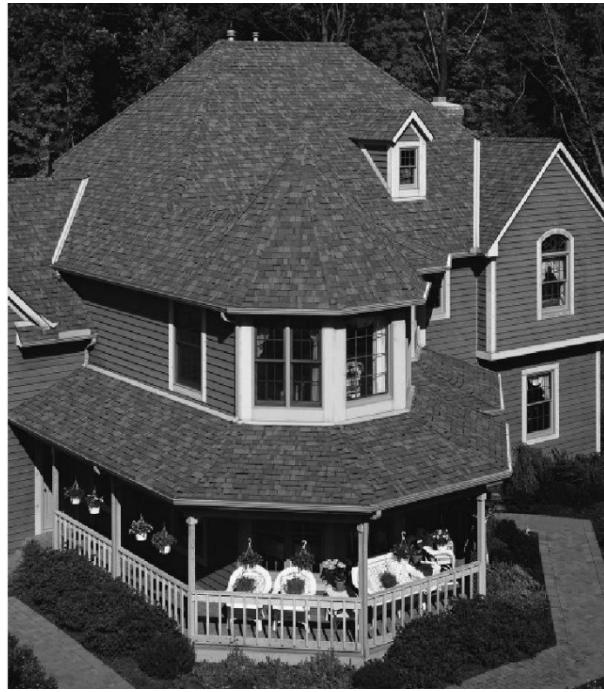
Rezultati funkcije su poređeni sa Matlab funkcijom imresize za metode najbližeg suseda i bikubične interpolacije, sa i bez filtriranja pre decimacije slike, pri čemu parametar s uzima vrednosti 2,4, i 7. Na slikama 2.1, 2.2 i 2.3 su prikazani rezultati za $s = 2$, $s = 4$, $s = 7$, respektivno.



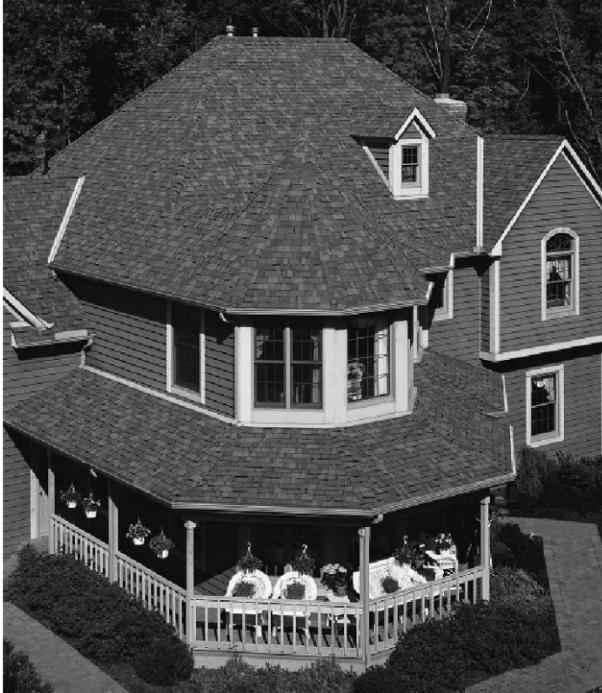
Slika 2.1.1 dos_downscale (*I, 2*)



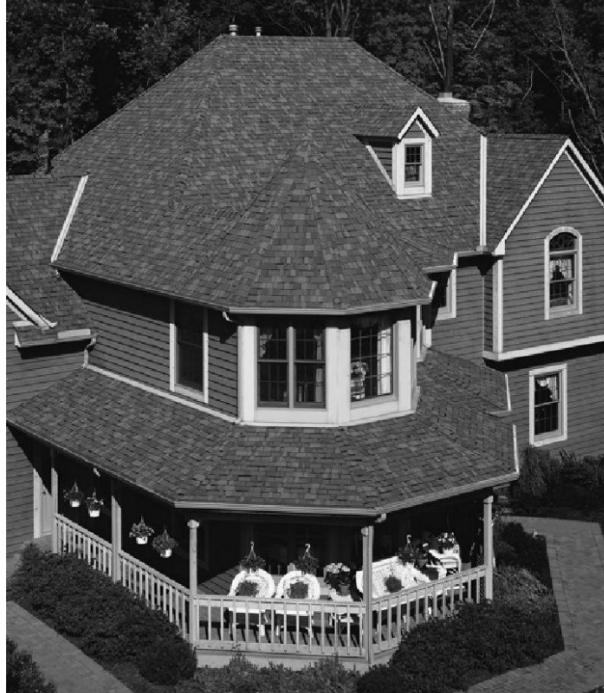
Slika 2.1.2.1. `imresize(I, 0.5 ,
'nearest', 'antialiasing', false)`



Slika 2.1.2.2. `imresize(I, 0.5 ,
'nearest', 'antialiasing', true)`



Slika 2.1.3.1. `imresize(I, 0.5 ,
'bicubic', 'antialiasing', false)`



Slika 2.1.3.2. `imresize(I, 0.5 ,
'bicubic', 'antialiasing', true)`

Kada je $s = 2$, razlika između decimacije slike sa i bez filtriranja je jako mala i jedino se pažljivim gledanjem slike primećuje aliasing efekat. Ovo se dešava zato što početna slika nema toliko jake

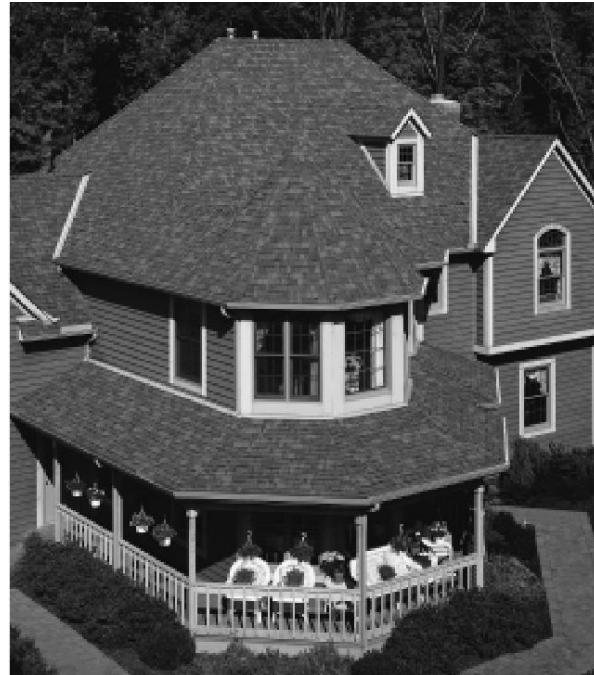
komponente na učestanostima koje su veće od odgovarajućeg f_{\max} , pa njihov doprinos ne dovodi to toga da se slika previše kvari. Kako smanjujemo s , doprinosi zbog preklapanja spektra su veći - već za $s = 4$, rezultati bez filtriranja postaju kritični, dok su za slučaj $s = 7$ gotovo neprihvatljivi.



Slika 2.2.1 dos_downscale(I, 4)



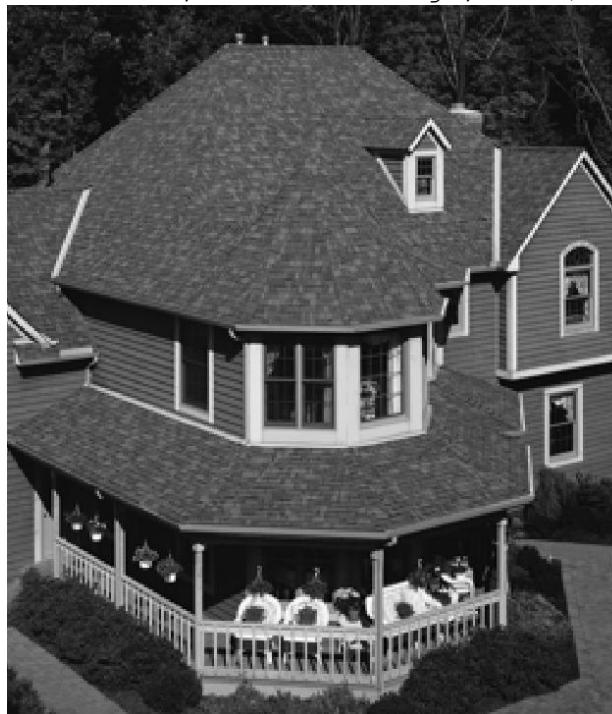
Slika 2.2.2.1. `imresize(I, 0.25 ,
'nearest', 'antialiasing', false)`



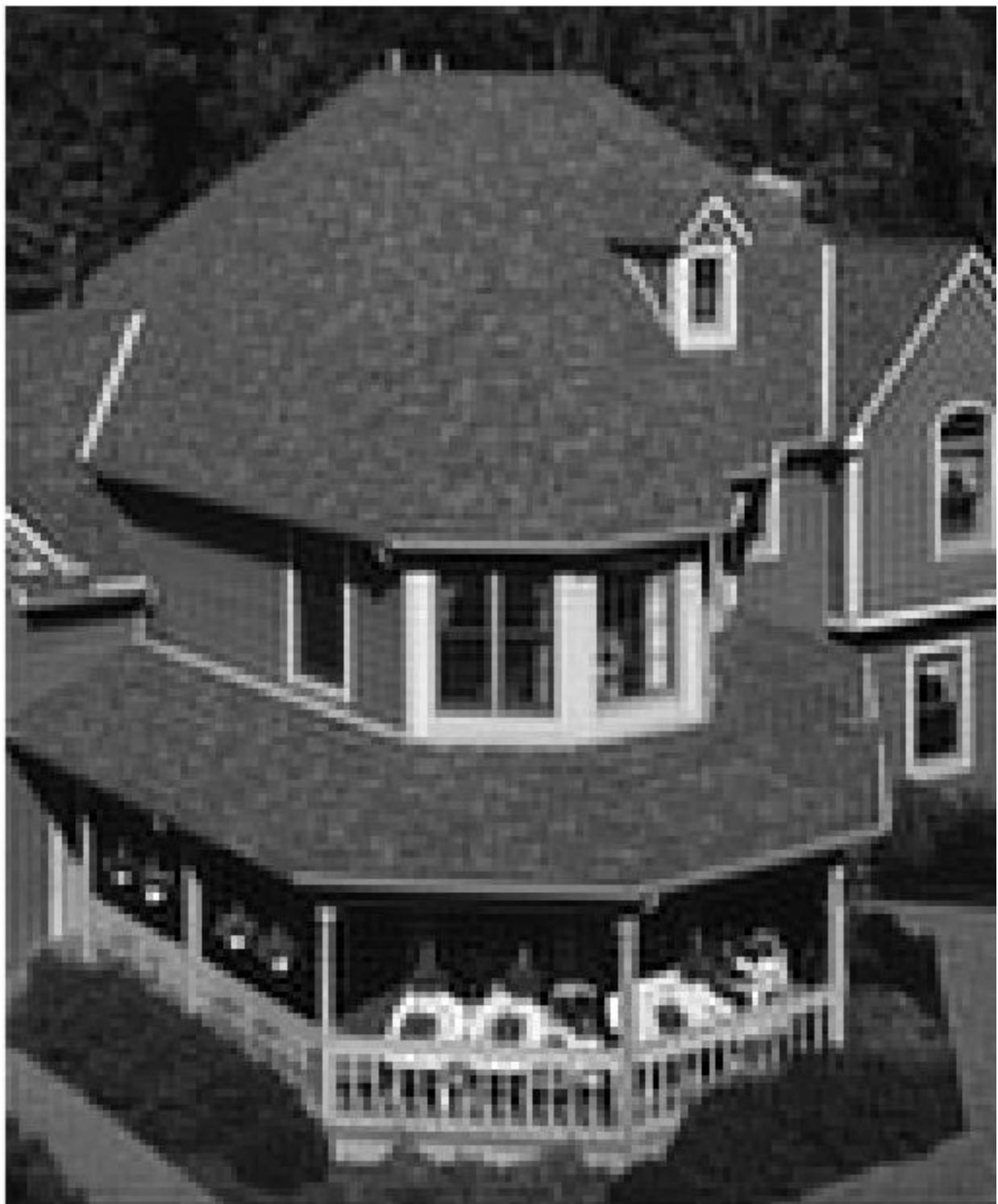
Slika 2.2.2.2. `imresize(I, 0.25 ,
'nearest', 'antialiasing', true)`



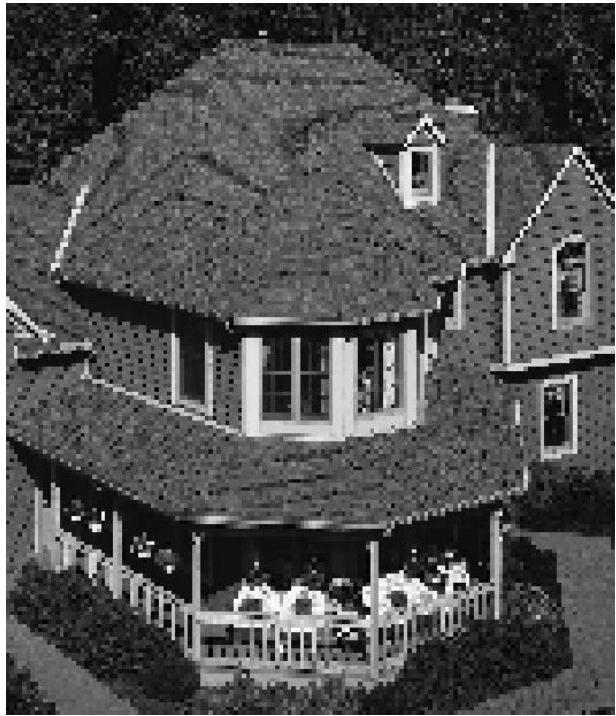
Slika 2.2.3.1. `imresize(I, 0.25 ,
'bicubic', 'antialiasing', false)`



Slika 2.3.3.2. `imresize(I, 0.25 ,
'bicubic', 'antialiasing', true)`



Slika 2.3.1 dos_downscale(I, 7)



Slika 2.3.2.1. `imresize(I, 1/7 , 'nearest', 'antialiasing', false)`



Slika 2.3.2.2. `imresize(I, 1/7 , 'nearest', 'antialiasing', true)`



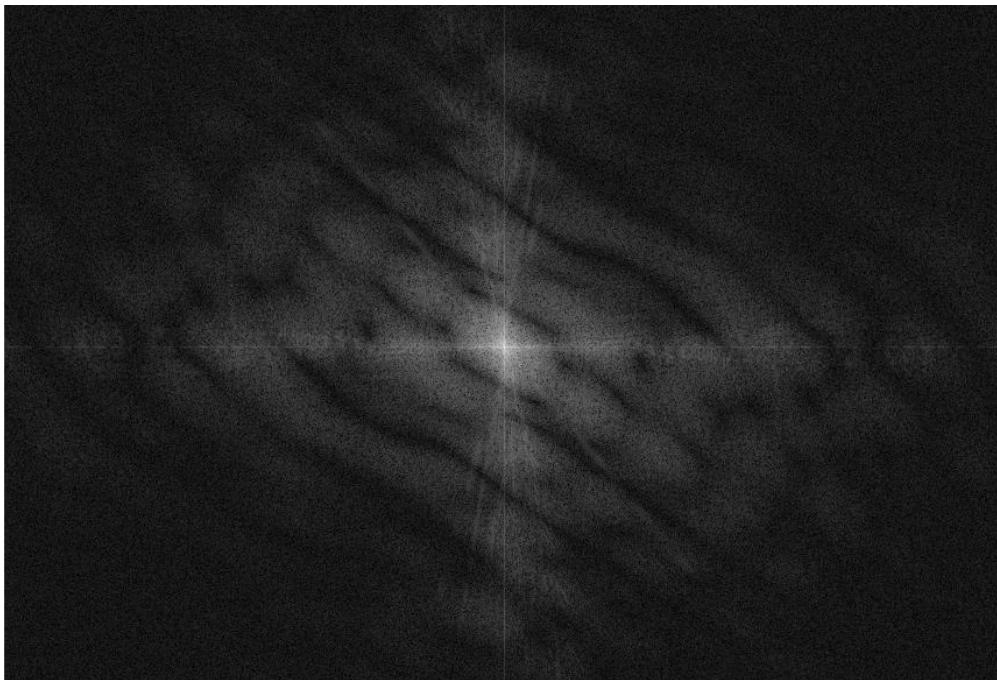
Slika 2.3.3.1. `imresize(I, 1/7 , 'bicubic', 'antialiasing', false)`



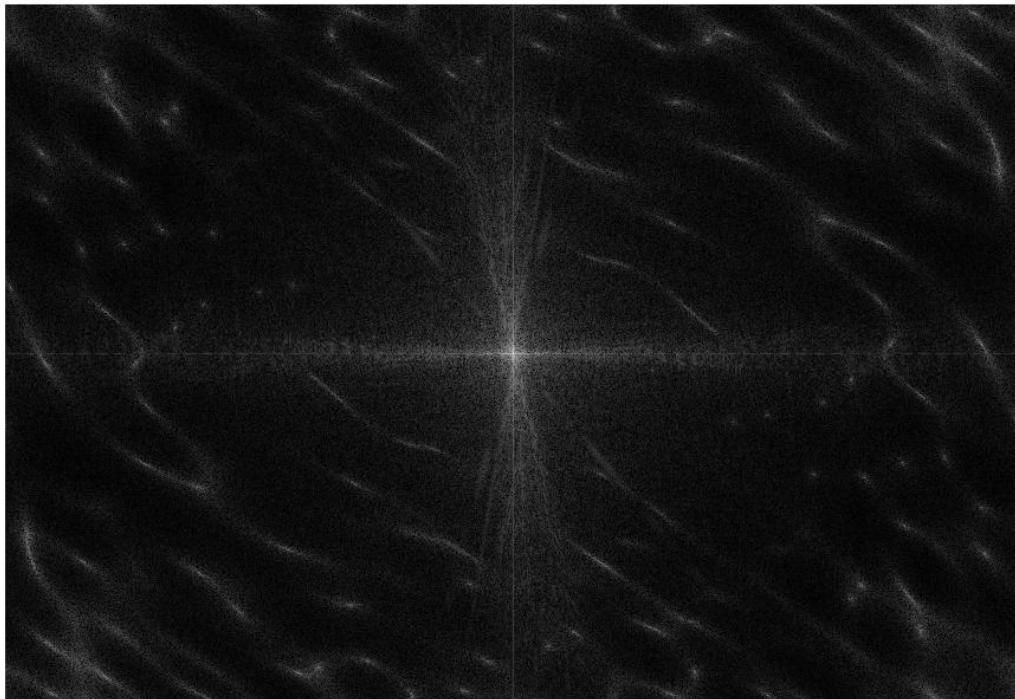
Slika 2.3.3.2. `imresize(I, 1/7 , 'bicubic', 'antialiasing', true)`

Restauracija slike

Restauracija je urađena korišćenjem Wiener-ovog filtra. Na slikama 3.1 i 3.2 je prikazan spektar slike pre i posle filtriranja. Na slici 3.3 se može videti rezultat primene ovog filtra nakon razvlačenja kontrasta, a na slici 3.4 histogram ove slike.



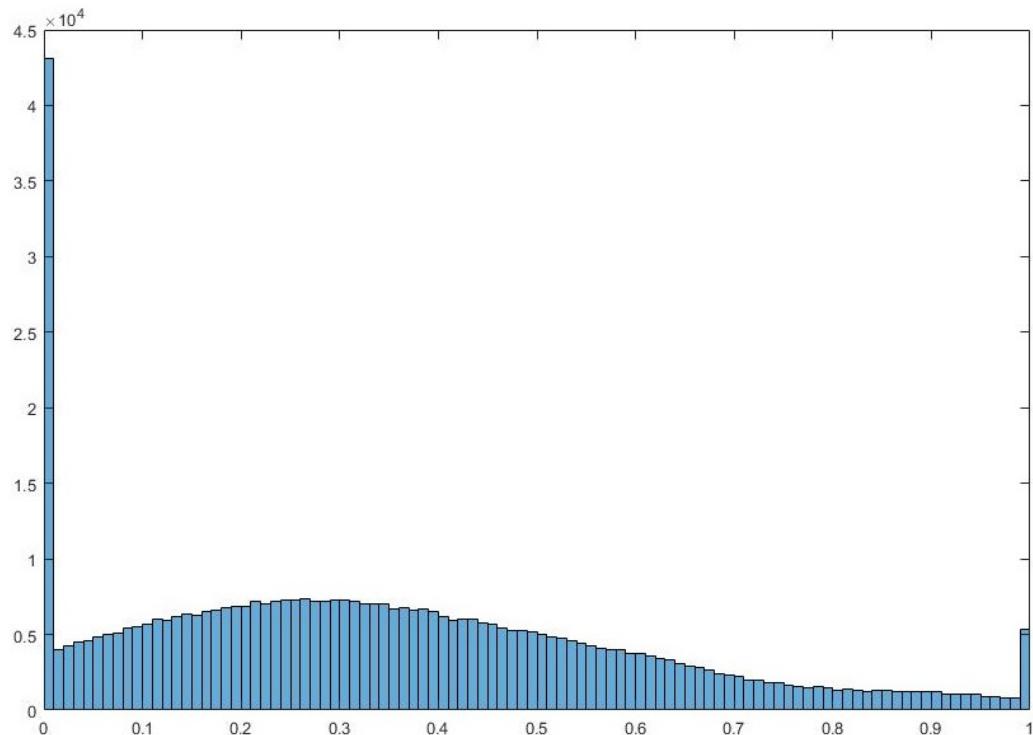
Slika 3.1. Spektar degradirane slike



Slika 3.2. Spektar slike nakon filtriranja Wiener-ovim filtrom



Slika 3.3 Slika nakon filtriranja i *imadjust* funkcije



Slika 3.4. Histogram slike nakon filtriranja i *imadjust* funkcije

Sa histograma se jasno vidi da postoji impulsni šum. Takođe, trebalo bi ukloniti linije sa desne i donje strane. One su posledica toga što je stvarna količina informacija manja od one koja se dobije kada dođe do pomeranja. Ispostavlja se da je prost medijan filter dovoljan za uklanjanje šuma.



Slika 3.5. Slika nakon primene medijan filtra



Slika 3.6. Slika nakon svih obrada

Funkcija kojom se realizuje ne-lokalno usrednjavanje slike

Funkcija **dos_non_local_means** (**I**, **K**, **S**, **var**, **h**) primenjuje algoritam ne-lokalnog usrednjavanja za uklanjanje šuma. Parametri koji se zadaju su, respektivno, slika, širina prozora nad kojim se proverava strukturalna sličnost, širina oblasti u okviru koje se pretražuju potencijalni kandidati za usrednjavanje, varijansa šuma i jačina odšumljivanja.

Neka je g slika koja se dobija nakon primene algoritma, f polazna slika. Neka je p piksel koji se trenutno poredi, a q piksel sa kojim se on poredi. Neka je K širina bloka centriranog oko piksela p , a S širina bloka oko piksela p iz koje se biraju pikseli q . Tada je vrednost piksela na novoj slici g data kao:

$$g(p) = \frac{1}{C(p)} \sum_{q \in S} f(q) w(p, q)$$

gde je $w(p, q)$ težinska funkcija.

$$C(p) = \sum_{q \in S} w(p, q)$$

$$w(p, q) = \exp\left(\frac{-\max(\|K(p) - K(q)\| - 2\sigma_n^2, 0)}{h^2}\right)$$

$$\|K(p) - K(q)\| = \frac{1}{|K(p)|} \sum_{t \in B(0)} (f(p+t) - f(q+t))^2$$

Algoritam se zasniva na ideji da ako neki pikseli imaju sličnu okolinu, to je verovatno zato što potiču od istog signala. Za svaki piksel p se proverava sličnost sa pikselima iz bloka čija je veličina definisana parametrom S . Sličnost dva piksela se opisuje težinskom funkcijom. Za date piksele p i q se računa RMS u bloku veličine B koji ih okružuje i ako je dva puta manja od zadate smatra se da pikseli pripadaju istom signalu.

Funkcija je naivna implementacija ovog algoritma. Testirana je za sve kombinacije K iz skupa { 3, 5, 7, 9, 11 } i S iz skupa { 15, 25, 33, 45, 52¹ }. Parametar h je 0.01, a $var = 0.0075$. Varijansa je određena korišćenjem Matlab funkcija **roi_poly** i **var**. Parametar h je određen iterativno za najmanje vrednosti K i S tako da se dobije najveća vrednost PSNR-a. Pored PSNR-a, praćeno je i vreme koje je potrebno da se funkcija izvrši.

Na osnovu algoritma se može proceniti složenost naivne implementacije ove funkcije, gde su M i N dimenzije slike:

$$O(MNBS) = MN S^2 K^2$$

Na slikama od 4.1. do 4.9. su prikazane slike koje se dobiju kada je $K = 3, 5, 9$ i $S = 15, 33, 52$. Na slici 4.10 se vidi zavisnost PSNR-a, a na slici 4.11 se vidi zavisnost vremena izračunavanja funkcije od ovih parametara. Na osnovu ovoga je i izabrane vrednosti za K i S koje se nalaze u glavnom programu.

¹ Trebalo je da bude 51, nažalost primećeno je tek kada se izvršen ceo program. Pošto je cilj bio naći zavisnost, ostavljena je vrednost 52.



Slika 4.1. $K = 3, S = 15$



Slika 4.2. $K = 3, S = 33$



Slika 4.3. $K = 3, S = 52$



Slika 4.4 $K = 5, S = 15$



Slika 4.5. $K = 5, S = 33$



Slika 4.6. $K = 5, S = 52$



Slika 4.7. $K = 9, S = 15$



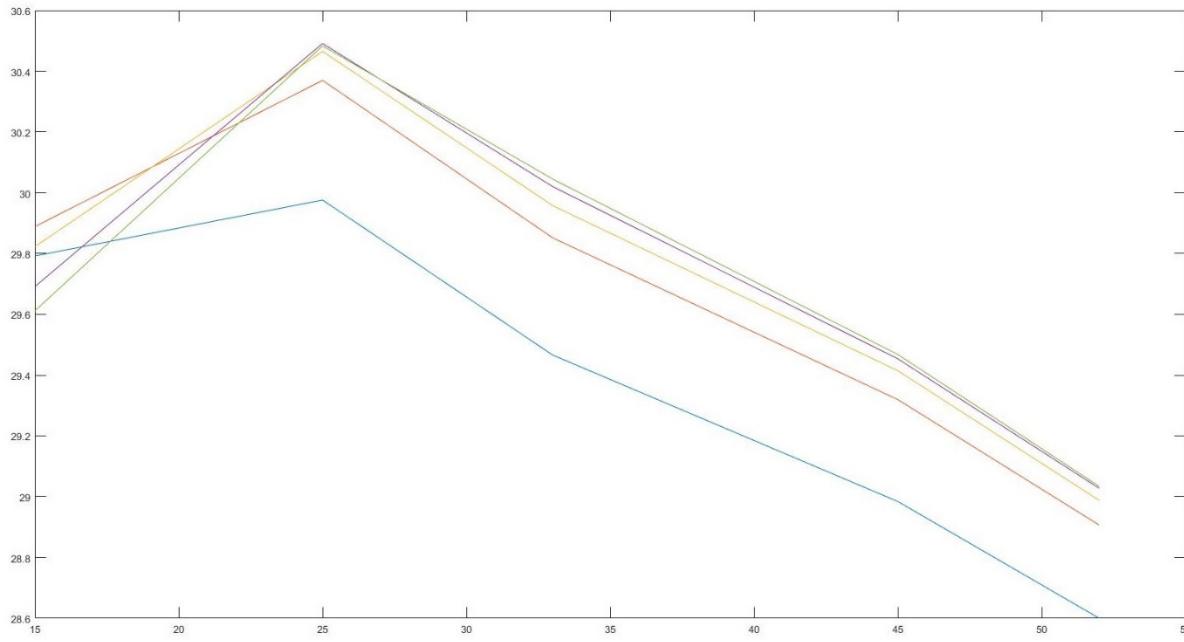
Slika 4.78 $K = 9, S = 33$



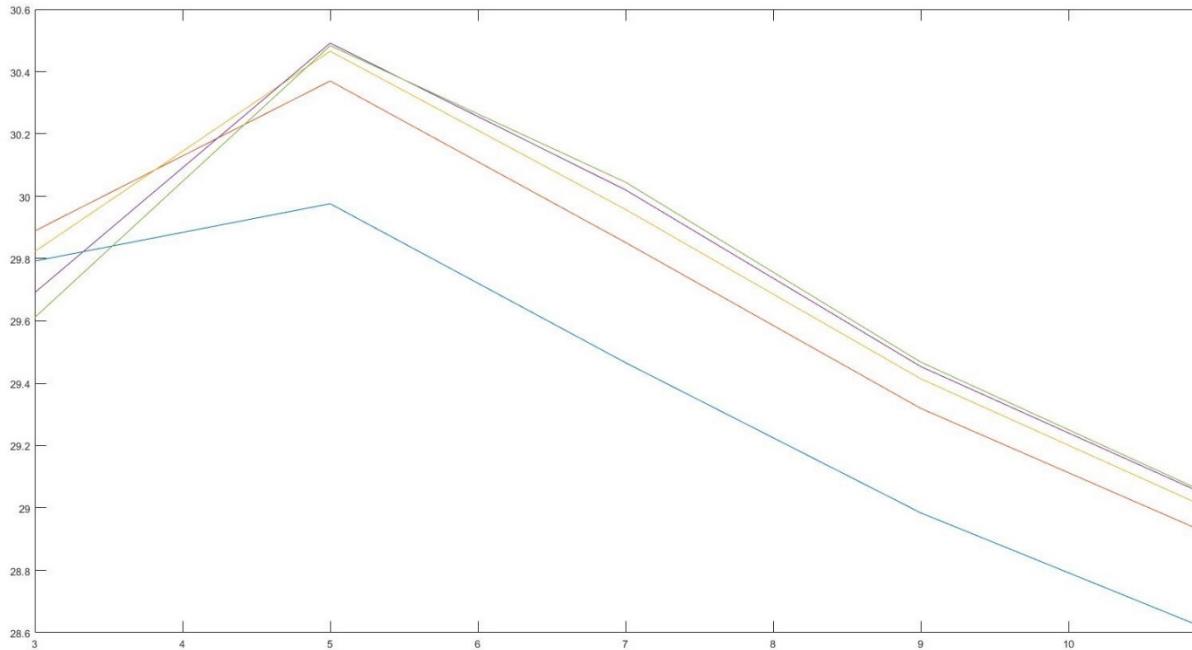
Slika 4.9. $K = 9$, $S = 52$



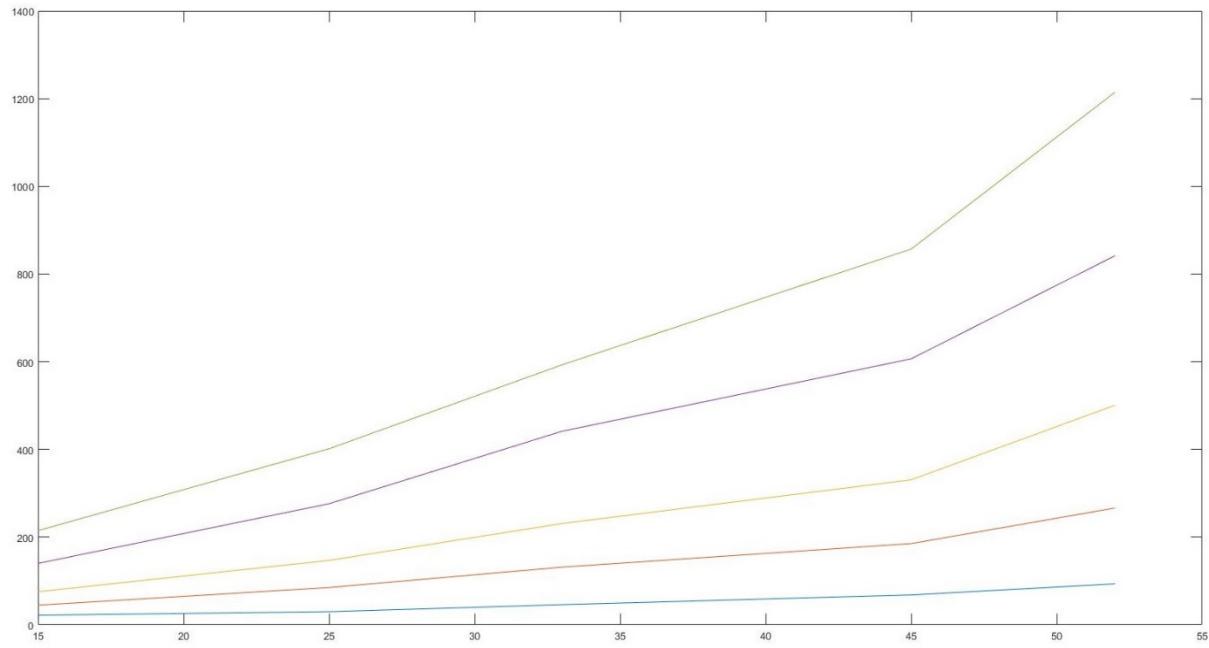
Slika 4.10. $K = 5$, $S = 25$ (izabrana slika)



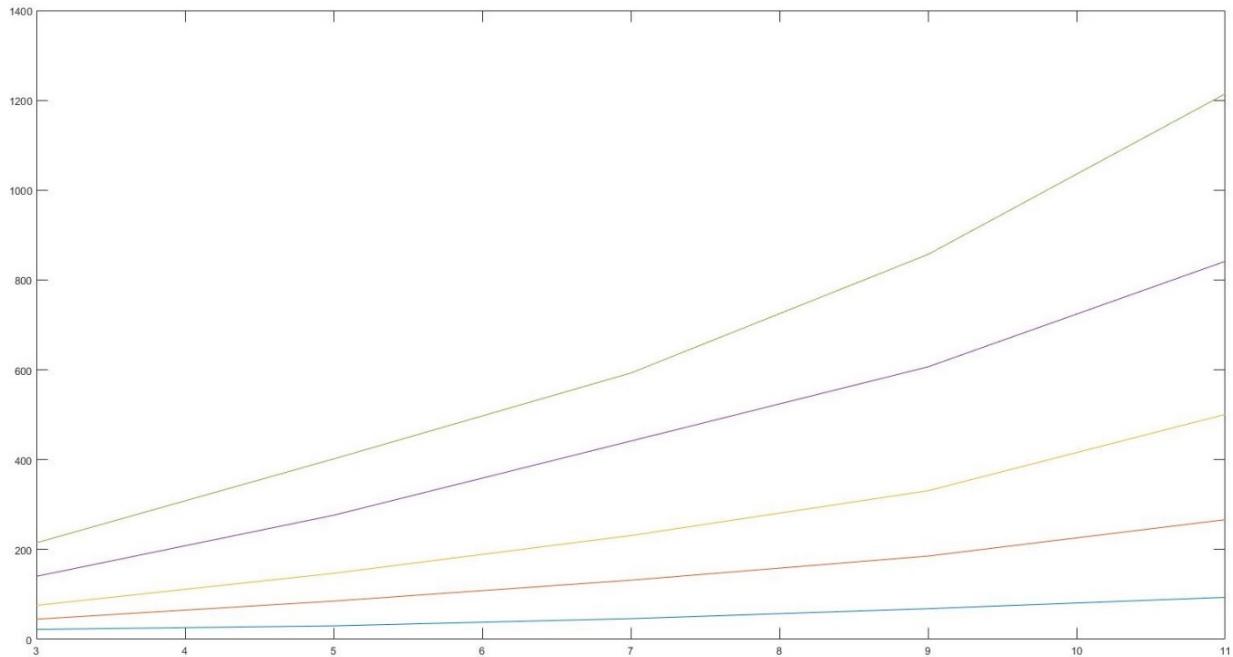
Slika 4.11.1. PSNR u funkciji S



Slika 4.11.2. PSNR u funkciji K



Slika 4.12.1. Zavisnost vremena izvršavanja od S



Slika 4.12.2. Zavisnost vremena izvršavanja od K

Obzirom da vreme izvršavanja kvadratno raste sa porastom S , uzeto je $S = 25$, $K = 5$. PSNR ima maksimalnu vrednost za $K = 5$, $S = 33$, ali je razlika zanemarljiva u odnosu na to kolika je ušteda u brzini izvršavanja.

Kôd kojim je određeno h:

```
%-----%
% Find optimal h value
% Take small values for S and K to speed it up
% Variance is found by using Matlab's roipoly() function
K = 5; S = 15; var = 0.0075;
h_start = 0.5;

|for p = 1:5
    %start with these values
    h = [0.1, 0.5, 0.8, 1, 1.5, 2]*h_start;

    %Compute image for each h
    D = cell(size(h));
    |for i = 1:size(h,2)
        D{i} = dos_non_local_means(I, K, S, h(i), var);
    end

    %calculate PSNR for each h
    PSNR_h = zeros(size(h));

    |for i = 1:size(h, 2)
        PSNR_h(i) = psnr(D{i}, original);
    end

    %take h for which PSNR is highest
    [maxPSNR, ind] = max(PSNR_h);
    h_start = h(ind);
end
%-----%
```

Kôd kojim je određen su testirane kombinacije za S i K:

```
%-----
%Compute dos_non_local_means for each combination of K and S
%Collect PSNR and computing time for each combination.

K = [3, 5, 7, 9, 11];
S = [15, 25, 33, 45, 52];
time = zeros(size(K,1), size(S,1));
PSNR = zeros(size(K,1), size(S,1));
G = {};
G{size(K,2)+1, size(S,2)+1} = [];

%Calculate dos_non_local_means for each combination of S and K
for i=1:size(K,2)
    for j=1:size(S,2)
        tStart = tic;
        G{i,j} = dos_non_local_means(I, K(i), S(j), var, h);
        time(i, j) = toc(tStart);
    end
end

%Calculate PSNR for each combination of S and K
for i=1:size(K,2)
    for j=1:size(S,2)
        PSNR(i,j) = psnr(G{i, j}, original);
    end
end

%Show results and corresponding parameters
for i=1:size(K,2)
    for j=1:size(S,2)
        figure; imshow(G{i,j});
        cS = num2str(S(j));
        cK = num2str(K(i));
        cPSNR = num2str(PSNR(i,j));
        title(strcat('S = ', cS, ' K = ', cK, ' PSNR = ', cPSNR));
    end
end
-----%
```