Usporedba odabranih metoda za uklanjanje multiplikativnog šuma iz digitalnih slika

Ercegovac, Luka

Sveučilište Jurja Dobrile u Puli

Tehnički fakultet

LUKA ERCEGOVAC

USPOREDBA ODABRANIH METODA ZA UKLANJANJE MULTIPLIKATIVNOG ŠUMA IZ DIGITALNIH SLIKA

Diplomski rad

Pula, rujan 2024. godine

Sveučilište Jurja Dobrile u Puli

Tehnički fakultet

LUKA ERCEGOVAC

USPOREDBA ODABRANIH METODA ZA UKLANJANJE MULTIPLIKATIVNOG ŠUMA IZ DIGITALNIH SLIKA

Diplomski rad

JMBAG: 0303092241, redoviti student

Studijski smjer: Računarstvo

Kolegij: Digitalna obrada i analiza slike

Znanstveno područje: Tehničke znanosti

Znanstveno polje: Računarstvo

Znanstvena grana: Obradba informacija

Mentor: doc. dr. sc. Nikola Lopac

Pula, rujan 2024. godine

Doc. dr. sc. Nikola Lopac
(Ime i prezime nastavnika)

Digitalna obrada i analiza slike
(Predmet)

Sveučilište Jurja Dobrile u Puli

Tehnički fakultet u Puli

ZADATAK TEME DIPLOMSKOG RADA

Pristupniku **MBS:**

Luka Ercegovac **0303092241**

Studentu Tehničkog fakulteta u Puli, izdaje se zadatak za diplomski rad – tema rada pod nazivom:

USPOREDBA ODABRANIH METODA ZA UKLANJANJE MULTIPLIKATIVNOG ŠUMA IZ DIGITALNIH SLIKA

Sadržaj zadatka:

U radu je potrebno usporediti odabrane metode za uklanjanje multiplikativnog zrnatog (*engl. speckle*) šuma iz digitalnih slika. Odabrane metode uklanjanja šuma, poput prostornog usrednjavanja, korištenja valića i nelokalnog usrednjavanja, potrebno je opisati, implementirati i primijeniti na skup testnih slika. Potrebno je provesti analizu dobivenih rezultata korištenjem mjera kvalitete slike poput vršnog odnosa signala i šuma (PSNR) i indeksa strukturalne sličnosti (SSIM). Potrebno je provesti evaluaciju i diskusiju učinkovitosti odabranih metoda u smanjenju šuma i očuvanju bitnih detalja slike.

Rad obraditi sukladno odredbama Pravilnika o završnom/diplomskom radu Sveučilišta u Puli.

Luka Ercegovac
(Ime i prezime studenta):

Redovni
(status izvanredni/redovni)

Diplomski sveučilišni studij Računarstvo
(studij)

Datum: 9. ožujka 2024.

Potpis nastavnika Nikola Lopac

Sadržaj

1. Uvod

Mnogi dijelovi opreme u zdravstvenim ili satelitskim odjelima oslanjaju se na digitalne slike, kao i stari fotoaparati. Jedna od najčešćih prepreka s kojima se digitalna slika suočava je postojanje raznih vrsta šumova, koji uništavaju jasnoću slike i čine daljnju analizu ili interpretaciju gotovo nemogućom U ovom radu raspravljat ćemo o problemu šuma u digitalnim slikama, posebice multiplikativnog šuma, te kako ga ukloniti sa slike.

Šum digitalne slike nasumična je varijacija informacija o svjetlini ili boji koja skriva važne značajke i smanjuje jasnoću slike. Multiplikativni šum obično se nalazi na slikama dobivenim korištenjem konkretnih tehnologija snimanja kao što su ultrazvuk, radar i lasersko snimanje. Multiplikativni šum je zrnast i predstavlja značajnu poteškoću u obradi slike zbog svoje multiplikativne prirode i komplicirane strukture.

Primarni cilj ovog rada je istražiti i evaluirati nekoliko pristupa za uklanjanje multiplikativnog šuma s digitalnih slika. Odabrane su tri metode za uklanjanje multiplikativnog šuma: metoda prostornog usrednjavanja (*engl. spatial averaging method*), metoda korištenja valića (*engl. wavelet method*) i metoda nelokalnog usrednjavanja (*engl. non-local means method*). Svaki od navedenih pristupa bit će predstavljen u smislu njihovih temeljnih načela, postavljanja parametara i implementacije.

Rezultati navedenih metoda uklanjanja šuma procijeniti će se korištenjem različitih testova izvedenih na odabranom skupu testnih slika. Rezultati će se zatim ispitati korištenjem dviju poznatih mjera kvalitete slike: vršnog odnosa signala i šuma (*engl. Peak Signal-to-Noise Ratio, PSNR*) i indeksa strukturalne sličnosti (*engl. Structural Similarity Index, SSIM*). Testiranje će se primarno usredotočiti na to koliko učinkovito metode smanjuju šum dok zadržavaju najbitnije aspekte slike.

Sljedeća poglavlja baviti će se postojanošću multiplikativnog šuma i odabranim metodama smanjenja multiplikativnog šuma, također odabrane metode će se implementirati u programskom jeziku Python i analizirati. Na kraju ćemo provesti evaluaciju i diskusiju o odabranim metodama, odnosno koliko dobro smanjuju šum uz očuvanje bitnih značajki slike.

Drugo poglavlje daje pregled šuma na digitalnim slikama, objašnjavajući osnovne vrste šuma i kako one utječu na kvalitetu slike. Također obrađuju se razlozi zbog koji šum nastaje i pod kojim uvjetima se pojavljuje.

Treće poglavlje fokusira se na zrnati šum, koji se često pojavljuje u slikama stvorenim pomoću koherentnih svjetlosnih izvora te onima dobivenim pomoću radarskih ili ultrazvučnih sustava. Objašnjava se matematički model zrnatog šuma, njegova multiplikativna priroda i kako on utječe na sliku.

Četvrto poglavlje predstavlja odabrane metode za uklanjanje zrnatog šuma iz digitalnih slika. Detaljno su opisane metrike za evaluaciju kvalitete slike, koje će se koristiti za procjenu učinkovitosti metoda.

Peto, šesto i sedmo poglavlje govore o jednoj od odabranih metoda za uklanjanje zrnatog šuma. U petom poglavlju obrađuje se metoda prostornog usrednjavanja, koja predstavlja osnovi pristup uklanjanja šuma. Šesto poglavlje bavi se metodom korištenja valića, koja koristi diskretnu valićnu transformaciju kako bi smanjila šum i zadržala ključne značajke slike. U sedmom poglavlju obrađuje se metoda nelokalnog usrednjavanja, koja koristi informacije iz cijele slike za uklanjanje šuma.

Zadnje osmo poglavlje prikazuje rezultate primjene svake od navedenih metoda na odabranom skupu slika. U prvom dijelu ovog poglavlja prvo se analiziraju rezultati svake metode posebno, a zatim se metode uspoređuju kroz korištenje metrika PSNR i SSIM. Zadnji dio ovog poglavlja prolazi kroz analizu metoda kada su one primijenjene na slikama gdje je zrnati šum prirodna pojava.

Cilj rada je pokazati prednosti i nedostatke različitih metoda za smanjenje multiplikativnog šuma, kao i pridonijeti napretku tehnologije obrade slike.

2. Šum na digitalnim slikama

Šum na slikama definiran je kao nasumična varijacija svjetline ili informacije o boji unutar slike, a najčešće je ta varijacija uzrokovana elektroničkim šumom. Obično se izraz „šum slike“ odnosi na šum u 2D slikama, a ne na 3D slikama. Šum mogu uzrokovati senzor slike i elektronika koja se nalazi u uređajima koji proizvode sliku, kao što su digitalne kamere. Također šum mogu uzrokovati uvjeti okoline i intrinzična fizička svojstva svjetla (Analytics India magazine PVT & AIM Media house LLC, 2024).

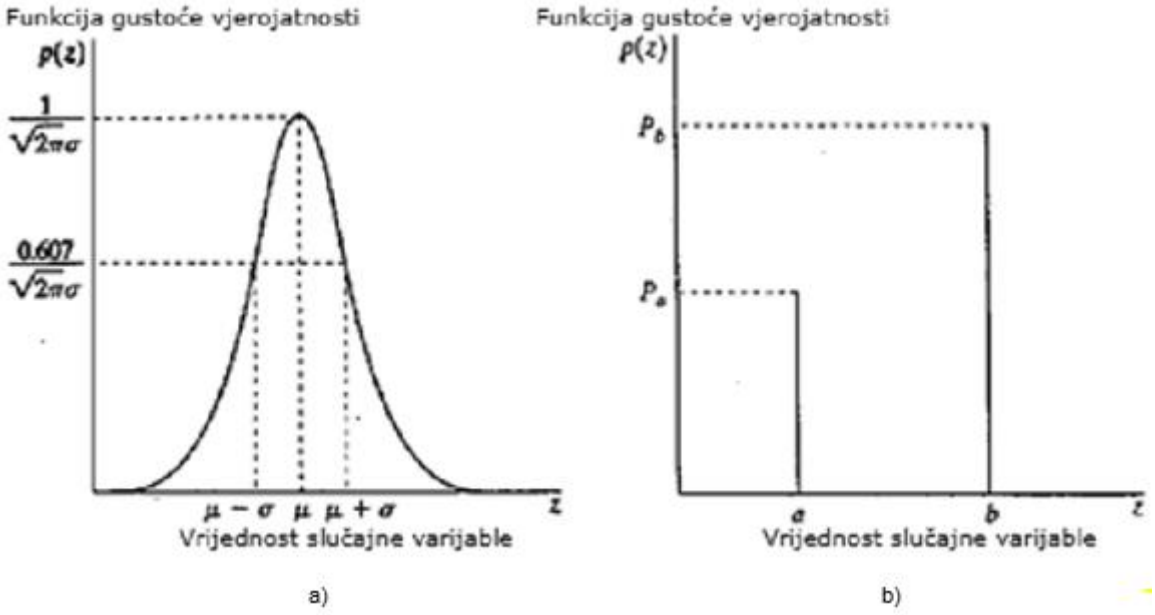
Kako slika izgleda kada je na njoj prisutan šum prikazano je na Slici 1.

Slika 1 – Prikaz slike sa šumom i bez šuma: a) Slika bez šuma, b) Slika sa šumom
Izvor: Izradio autor

Kako možemo znati da na slici imamo šum? Šum na slici je najčešće u obliku točkica i zrnaca. Slika 1 a) prikazuje izvornu sliku bez šuma, a Slika 1 b) istu sliku uz prisutnost šuma, pri čemu se može jasno razaznati i primijetiti kako šum utječe na kvalitetu slike. Na Slici 1 primjetno je da šum na sliku dodaje neželjene informacije, što rezultira neželjenim ishodima kao što su izobličenja, iskrivljeni rubovi, prikrivene linije i kutovi, zamućeni objekti i poremećena pozadina. Također u nekim situacijama šum može uzrokovati poremećaj u boji na slici, a ako ga ima previše može odvraćati pozornost od glavnog djela slike. Dvije najčešće vrste šuma na slici su Gaussov šum i šum salt-and-pepper. Osim ove dvije najčešće vrste postoji još puno vrsta šuma, a među njima je i šum koji ćemo kasnije obraditi - multiplikativni šum (*engl. speckle noise*).

Za opisivanje statističke distribucije šuma koristi se funkcija gustoće vjerojatnosti (*engl. Probability Density Function, PDF*). Svaki šum ima svoju karakterističnu

distribuciju koja je predstavljena funkcijom gustoće vjerojatnosti, tako i dva najčešća šuma Gaussov šum i šum salt-and-pepper, imaju različitu karakterističnu distribuciju. Gaussov šum ima distribuciju oblika zvona, a razlog tome je jer on mijenja vrijednost sive boje na slici, dok se šum salt-and-pepper pojavljuje kao nasumični bijeli i tamni pikseli i njegov PDF izgleda kao ravna linija sa skokovima na minimalnim i maksimalnim vrijednostima svjetline, što prikazuje Slika 2 (Boyat & Joshi, 2015).



Slika 2 – PDF: a) Gaussovog šuma, b) šuma salt-and-pepper
Izvor: Autor modificirao prema (Boyat & Joshi, 2015)

Slika 2 a) prikazuje statističku distribuciju za Gaussov šum, Slika 2 b) prikazuje statističku distribuciju za šum salt-and-pepper. Osim ova dva šuma multiplikativni šum također ima svoju statističku distribuciju koja je obično eksponencijalna distribucija. Razumijevanje statističke distribucije šuma omogućava nam da lakše osmislimo načine kako da smanjimo taj šum na slici. Iako je šum u nekoj mjeri uvijek prisutan, nekoliko je načina kako možemo smanjiti šum na slikama prilikom njihovog prikupljanja:

- Podešavanje postavki kamere: podešavanje osjetljivosti kamere na svijetlo, niža osjetljivost je bolja u situacijama kada je svjetlost dobra, a viša kada je manje osvjetljenje.
- Stabilizacijom kamere: korištenjem alata kao što su tronožac, kako se kamera ne bi previše tresla
- Format snimanja: korištenjem RAW formata, jer on bilježi sve podatke senzora

- Izbjegavanje prevelike topline: prevelika toplina fotoaparata ili upotrebe u vrućim okruženjima doprinosi stvaranju šuma (Adobe, 2024).

Osim što se šum može smanjiti kada se slike prikupljaju, on se može smanjivati i upotrebom određenih alata kao što su uređivači slika, specijalizirani softveri za smanjenje šuma ili određenih metoda / filtra za uklanjanje šuma. Izbor metode / filtra za uklanjanje šuma ovisi o samom šumu, o količini šuma na slici i o željenom rezultatu. Postoji puno metoda za uklanjanje šuma, ali nisu sve jednako učinkovite na svakom šumu, neke su bolje za određeni šum, a neke lošije.

Tri metode koje ćemo obraditi u radu su metoda prostornog usrednjavanja, metoda korištenja valića i metoda nelokalnog usrednjavanja. Pritom svaka metoda ima svoje prednosti i nedostatke. Iako je smanjivanje šuma poželjno ono može imati neočekivane nuspojave. Tako neke metode umjesto da smanje šum i poboljšaju kvalitetu slike, mogu je dodatno pogoršati. Najčešće nuspojave smanjivanja šuma su zamućenje slike i gubitak bitnih detalja na slici. Ako se metode koriste na ispravan način i na šumu koji nabolje smanjuju mogu smanjiti šum, poboljšati kvalitetu slike, poboljšati detalje na slici i poboljšati kontrast boja.

3. Zrnati (*engl. speckle*) šum

Multiplikativni šum ili još poznat i kao zrnati šum je zrnasti uzorak koji se pojavljuje na slikama i signalima koje generira koherentna svjetlost ili neki drugi izvori valova. Budući da svjetlosni valovi iz svake točke na površini prelaze različite udaljenosti i mogu utjecati jedni na druge i tako ili stvarati svijetle točke ili tamne točke, zbog čega slika izgleda mutno ili nejasno (Singh & Pandey, 2016).

Zrnati šum proizlazi iz koherentnog karaktera izvora svjetlosti. Na primjer, laserski izvor svjetlosti ima sinkronizirane svjetlosne valove. Kada se ti valovi sudare s hrapavom površinom ili velikim brojem čestica, raspršuju se i interferiraju jedni s drugima, tvoreći svijetle i tamne točke. Na zrnatost utječe hrapavost površine i valna duljina izvora svjetlosti, što znači da ako je veća hrapavost površine i manja valna duljina to će zrnatost šuma biti veća (Singh & Pandey, 2016).

Zrnati šum je problem koji se pojavljuje na slikama koje koriste tehnologije sa koherentnom svjetlosti, što uključuje:

- Lasersko skeniranje: pojavljuje se kod korištenja LiDAR-a u autonomnim automobilima i softveru za 3D mapiranje. Raspršenost zrnatog šuma uzrokuje bljesak svjetlosti različitih intenziteta, a to otežava mjerenje udaljenosti i generiranje 3D modela (Dan & Zhimin, 2019).
- Radarsko snimanje: zrnati šum uzrokuje zamućenje slike, te rubovi objekta na slici možda neće biti vidljivi (Gonzalez, 2023).
- Medicinsko snimanje: kod medicinskih slika od kojih su neke same po sebi teške za razumjeti i očitati, zrnati šum pogoršava tu situaciju (Gonzalez, 2023).

Zrnati šum može jako utjecati na kvalitetu slike. Za razliku od drugih šumova, zrnati šum ima multiplikativnu karakteristiku. To znači da zrnati šum množi izvorni signal sa slučajnom varijablom, iz čega možemo zaključiti da će svjetliji dijelovi slike imati uočljiviju varijaciju točkica. Zrnasti šum se na slici pojavljuje kao zrnati uzorak koji podsjeća na šum salt-and-pepper, ali s jako svijetlim i tamnim točkama koje su nasumično raspoređene. Kada je zrnati šum prisutan, ona gubi delikatne detalje i karakteristike. Teško je razlikovati objekte na slici, a sadržaj slike je teško razumjeti. Također zrnati šum može smanjiti kontrast slike tako što briše razlike u intenzitetu (Singh & Pandey, 2016).

Slika 3 prikazuje kako izgleda slika kada je na njoj prisutan zrnati šum.

Slika 3 – Prikaz slike bez šuma i sa zrnatim šumom: a) Slika bez šuma, b) Slika sa zrnatim šumom
Izvor: Izradio autor

Slika 3 a) prikazuje sliku bez ikakvog šuma, a Slika 3 b) prikazuje istu sliku s dodanim zrnatim šumom. Na slici s dodanim zrnatim šumom mogu se primijetiti svijetle i tamne točke i također teško je razaznati neke objekte na slici, na primjer kamere ili građevine u pozadini. Sa Slike 3 može se primijetiti da zrnati šum značajno smanjuje kvalitetu slike dodavanjem neželjenih promjena intenziteta koji zamućuju određene informacije. Takve promjene mogu rezultirati otežavanjem analize sadržaja slike, pogotovo kada je u pitanju prepoznavanje objekata i izdvajanje značajki. Izmjene u intenzitetu mogu povećati netočnost u mjerenjima udaljenosti ili visine, na primjer kod korištenja LiDAR-a ili na radarskim slikama. Također, segmentacija slike je otežana zbog nejasnoća u granicama koje dijele jedan dio slike od drugoga. Primjerice u dijelu Slike 3 koji uključuje šum, teže je raspoznati gdje završava rukav kaputa i počinje rukavica koju nosi osoba.

3.1. Matematički model zrnatog šuma

Model šuma se koristi za uklanjanje nepravilnosti na slici uzrokovane od strane šuma. Mjerenja poput lokalne varijance i srednje vrijednosti pomaže u prepoznavanju pripada li piksel značajki detalja slike ili dolazi iz nekog drugog izvora. Kada je u pitanju zrnati šum, gdje je razina omjera varijance i srednje vrijednosti viša od razine šuma, piksel je dio detaljne značajke slike i neće biti izmijenjen. Ako je omjer nizak, to znači da se

nalazi u homogenom području piksela i može se izmijeniti kako bi se smanjio šum (Singh & Pandey, 2016).

Zrnati šum se obično klasificira kao multiplikativni šum. Rezultat toga je da je izlazni signal kombinacija izvornog signala i zrnatog šuma. Neka nam $I(i,j)$ označava degradirani piksel promatrane slike, a $S(i,j)$ piksel slike bez šuma koji treba rekonstruirati. Multiplikativni model izgledao bi ovako (Singh & Pandey, 2016):

$$I(i,j) = S(i,j) \cdot N(i,j) \tag{1}$$

gdje nam $N(i,j)$ predstavlja multiplikativni šum s jediničnom srednjom vrijednosti i standardnom devijacijom.

Zrnati šum nije jednostavan, a matematički modeli dostupni su u određenoj mjeri za predstavljanje njegovog ponašanja. Odziv zrnatog šuma obično se prikazuje u obliku jednostavnog modela (Singh & Pandey, 2016):

$$y = x \cdot n \tag{2}$$

gdje je y složeni točkasti koeficijent raspršenja, x je izvorni netočkasti koeficijent raspršenja, a n je multiplikativni doprinos točkastog raspršenja. Model zrnatog šuma može se proširiti na polarimetrijski slučaj uzimajući u obzir da na polarimetrijske kanale utječu neovisne multiplikativne komponente (Singh & Pandey, 2016):

$$\begin{bmatrix} S_{11} \\ S_{12} \\ S_{22} \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & \cdots \\ \cdots & x_{12} & \cdots \\ \cdots & \cdots & x_{22} \end{bmatrix} \begin{bmatrix} n_{11} \\ n_{12} \\ n_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} |S_{11}|^2 \\ |S_{12}|^2 \\ |S_{22}|^2 \end{bmatrix} = \begin{bmatrix} X_{11}(n_{11}{n_{11}}^*) \\ X_{12}(n_{12}{n_{12}}^*) \\ X_{22}(n_{22}{n_{22}}^*) \end{bmatrix} \tag{3}$$

Polarimetrijsko snimanje je snimanje slika s različitim polarizacijama svjetlosti (ASU News, 2023).

4. Metode za uklanjanje zrnatog šuma iz digitalnih slika

Uklanjanje zrnatog šuma bitan je proces u različitim aplikacijama za obradu slike. Uklanjanjem zrnatog šuma možemo postići bolju kvalitetu slike, što nam olakšava analizu i obradu slike, te je vizualno lakša za razumjeti. Postoji više metoda za uklanjanje zrnatog šuma sa slike, a u ovom radu obradit će se:

- Metoda prostornog usrednjavanja
- Metoda korištenja valića
- Metoda nelokalnog usrednjavanja

Ove metode su odabrane jer svaka ima drugačiji pristup rješavanja problema, odnosno uklanjanja zrnatog šuma sa slike. Sve tri metode biti će implementirane u programskom jeziku Python. Nakon implementacije metoda, svaka će biti ispitana pomoću metrika za ocjenjivanje kvalitete rekonstrukcije slike. Korištene metode za ocjenjivanje kvalitete su: metrika vršnog odnosa signala i šuma (PSNR) i indeksa strukturalne sličnosti (SSIM).

4.1. Metrika vršnog odnosa signala i šuma (PSNR)

PSNR izračunava omjer između najveće moguće vrijednosti piksela na slici i srednje kvadratne vrijednosti razlike (*engl. Mean Squared Error, MSE*) između izvorne i rekonstruirane slike. Prednosti PSNR-a su to što je jednostavan i lak za izračunavanje i, budući da se bazira na MSE-u, neovisan je o smjeru razlike signala, odnosno izvorni ili rekonstruirani signal može se oduzeti jedan od drugoga i dati isti PSNR rezultat. Ograničenost PSNR-a je što se fokusira na razlike između piksela, zanemarujući vizualnu percepciju, jer slike koje imaju sličan PSNR neće biti slične ljudskom oku. Također je osjetljiv na velike količine šuma, što znači da jedan piksel sa značajnom razlikom u intenzitetu može značajno smanjiti rezultat PSNR-a (National instruments corp , 2024). Računanje PSNR-a izvodi se (National instruments corp , 2024):

$$PSNR = 20log_{10}\left(\frac{MAX_f}{\sqrt{MSE}}\right) \tag{4}$$

gdje je MAX_f maksimalna moguća vrijednost piksela (255 za 8-bitnu sliku). MSE računa se kao (National instruments corp , 2024):

$$MSE = \frac{1}{mn}\sum_{0}^{m-1}\sum_{0}^{n-1}\left||f(i,j) - g(i,j)|\right|^2 \quad (5)$$

Gdje su m i n dimenzije slike, a $f(i,j)$ i $g(i,j)$ su vrijednosti piksela na poziciji (i,j)

4.2. Metrika indeksa strukturalne sličnosti (SSIM)

SSIM metrika računa strukturalnu sličnost između izvorne i rekonstruirane slike, pritom razmatrajući svjetlinu, kontrast i sličnost strukture između odgovarajućih područja slike. Za razliku od PSNR-a, SSIM je bolji jer održava ljudsku percepciju. Također, za razliku od PSNR-a na kojeg može snažno utjecati jedan piksel, SSIM je otporniji na takve izolirane velike količine šuma. Nedostatak mu je što je u usporedbi s PSNR-om računanje SSIM-a računalno kompliciranije i potrebno je više vremena. Iako rezultat SSIM-a pruža dobar pokazatelj strukturne sličnosti, pojedinačne komponente ne daju jasnu interpretaciju kao jedna vrijednost PSNR-a (Wikipedia, 2024). SSIM se računa kao (Wikipedia, 2024):

$$SSIM(x,y) = l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma} \quad (6)$$

gdje su α, β i γ parametri za postavljanje relativne važnosti tri komponente. Obično je ta vrijednost jednaka 1 pa izraz izgleda ovako (Wikipedia, 2024):

$$SSIM = l(x,y) \cdot c(x,y) \cdot s(x,y) \quad (7)$$

gdje x i y predstavljaju određeni dio iz izvorne slike i rekonstruirane slike. Uzimajući u obzir da SSIM koristi svjetlinu, kontrast i sličnost strukture za izračun, postoje i formule za izračun pojedinačnog djela i one su (Wikipedia, 2024):

Za svjetlinu:

$$l(x,y) = \frac{2\mu_x\mu_{y+}C_1}{\mu_x^2 + \mu_y^2 + C_1} \quad (8)$$

gdje su μ_x i μ_y prosječni intenziteti slike x i y, a C_1 je konstanta za stabilizaciju dijeljenja sa slabim nazivnikom.

Za kontrast:

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_x^2 + C_2} \tag{9}$$

gdje su σ_x i σ_y standardno odstupanje od x i y, a C_2 je konstanta za stabilizaciju dijeljenja sa slabim nazivnikom.

Za strukturnu usporedbu:

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{10}$$

gdje je σ_{xy} kovarijanca od x i y.

C_1 i C_2, konstante za stabilizaciju računaju se kao (Wikipedia, 2024):

$$C_1 = (k_1 L)^2 \tag{11}$$

$$C_2 = (k_2 L)^2 \tag{12}$$

gdje L predstavlja dinamički raspon vrijednosti piksela, a ona obično iznosi:

$$2^{bitovi\ po\ pikselu} - 1 \tag{13}$$

k_1 i k_2 su zadane vrijednosti i iznose: $k_1 = 0.01$ i $k_2 = 0.03$. Treća konstanta za stabilizaciju C_3 računa se kao:

$$C_3 = \frac{C_2}{2} \tag{14}$$

Množenjem svih komponenti dobijemo krajnju formulu za SSIM. Raspon za rezultat je od -1 do 1. Ako je rezultat 1 to znači da je slika identična, ako je rezultat 0 znači da ne postoji sličnosti između slika, a ako je rezultat -1 to označava savršenu antikorelaciju (Wikipedia, 2024).

Kako bi mogli koristiti metode za uklanjanje zrnatog šuma i na kraju metode evaluirati metrikama, potrebna nam je implementacija dodavanja šuma na sliku. Za svaku metodu na sliku smo dodali šum pomoću biblioteke skimage i također iz iste biblioteke iskoristili smo funkcije za računanje metrika. Formula kako se dodaje zrnati šum na sliku je (scikit-image team, 2024):

$$šum = slika + n \cdot slika \tag{15}$$

gdje n predstavlja Gauussov šum.

5. Metoda prostornog usrednjavanja

Metoda prostornog usrednjavanja može se smatrati i najjednostavnijom i najpoznatijom metodom za prostorno filtriranje. Ova metoda može se koristiti kao:

- Metoda za smanjivanje šuma na slici
- Preliminarni korak obrade za učinkovitije funkcioniranje naprednih metoda

Metoda prostornog usrednjavanja funkcionira tako da se jednaka vrijednost w_K dodaje svim pikselima u susjedstvu oko odabranog piksela. Vrijednost koja se dodaje definira se kao

$$w_K = \frac{1}{NM} \tag{16}$$

gdje su $N \cdot M$ veličina susjedstva. Obično se u metodi prostornog usrednjavanja koristi konvolucija s 3x3 maskom čiji koeficijenti imaju vrijednost 1, a on se dijeli s faktorom skaliranja koji je jednak ukupnom broju elemenata u maski (Lopac, 2022). Primjer:

$$w(i,j) = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{17}$$

Iznad je prikaz konvolucijske maske za 3x3 susjedstvo kod korištenja metode prostornog usrednjavanja. U ovom slučaju faktor skaliranja je 9 i iz toga dobijemo da vrijednost koja se dodaje svakom pikseli iznosi 1/9. Osim ove jednostavne izvedbe metode prostornog usrednjavanja postoje i poboljšane izvedbe:

- Poboljšani koeficijent maske
- Usmjereno usrednjavanje
- Selektivna primjena rezultata izračuna prosjeka
- Uklanjanje ekstrema prije izračuna prosjeka (Lopac, 2022)

Prednosti metode prostornog usrednjavanja bi bile:

- Jednostavna implementacija
- Smanjenje šuma: smanjuje nasumične varijacije intenziteta koje je uzrokovao šum i izglađuje sliku.

- Preliminarni korak: može se koristiti za omogućavanje veće učinkovitosti naprednijih metoda (Lopac, 2022).

Nedostatci metode su:

- Nije prikladna za svaki šum: manje je učinkovita za šumove s odstupanjima.
- Zamućenje rubova: rubovi često imaju oštre prijelaze intenziteta, a ova metoda ih izglađuje i gube se detalji (Lopac, 2022).

5.1. Implementacija

Implementacija metode prostornog usrednjavanja napravljena je u programskom jeziku Python. Cijeli kod sastoji se od nekoliko funkcija kao što su: funkcija za biranje slike, funkcija za dodavanje zrnatog šuma na sliku, funkcija za uklanjanje šuma metodom prostornog usrednjavanja, funkcije za izračun metrika PSNR i SSIM, te glavna funkcija koja poziva sve prijašnje funkcije i prikazuje rješenje.

Slika 4 prikazuje uvoz svih potrebnih biblioteka.

```
import numpy as np
import cv2
from tkinter import Tk, filedialog
from matplotlib import pyplot as plt
from skimage.util import random_noise
from skimage.metrics import structural_similarity as ssim,
peak_signal_noise_ratio as psnr
import time
```

Slika 4 - Uvoz biblioteka za implementaciju svih metoda
Izvor: Izradio autor

Jedan od važnijih dijelova implementacije su biblioteke koje će nam biti potrebne kako bi implementacija bila što bolja i u potpunosti odrađena. Prva biblioteka `numpy` općenito se u Pythonu koristi za numeričko računanje jer ima podršku za nizove, matrice i mnoge druge matematičke funkcije (PyData Sphinx Theme, 2024). `Cv2` je biblioteka pod nazivom `OpenCV` koja je javno dostupna biblioteka za računalni vid i strojno učenje (GeeksforGeeks, 2024). Nadalje `tkinter` biblioteka se u Pythonu koristi za stvaranje grafičkog korisničkog sučelja. Iz te biblioteke iskoristili smo klasu `Tk` koja se koristi za stvaranje prozora aplikacije i modul `filedialog` koji se koristi za

otvaranje i spremanje datoteka putem grafičkog sučelja (Python Software Foundation, 2024). Biblioteka `matplotlib` koristi se za stvaranje statičnih, interaktivnih i animiranih vizualizacija. Iz te biblioteke, modul `pyplot` koristi se za izradu grafikona i vizualizaciju podataka (Matplotlib development team, 2024). Uvezli smo funkciju `random_noise` iz biblioteke `skimage.util` koja sadrži algoritme za obradu slike, s kojima smo slici dodali zrnati šum. `Skimage.metrics` biblioteka sadrži algoritme za računanje metrike, pa su tako iz nje dodane funkcije `structural_similarity` za računanje SSIM-a i funkcija `peak_signal_to_noise_ratio` za računanje PSNR-a. Sve biblioteke će se koristiti i u implementacijama ostalih metoda uklanjanja zrnatog šuma (scikit-image team, 2024).

Slika 5 prikazuje prvu bitnu funkciju, a koja je funkcija za učitavanje slike na kojoj će se metoda testirati.

```
def choose_image():
    Tk().withdraw()
    file_path = filedialog.askopenfilename(filetypes=[("Image files",
"*.jpg;*.jpeg;*.png;*.bmp;*.tif;*.gif;*.tiff")])
    return file_path
```

Slika 5 - Funkcija za učitavanje slike
Izvor: Izradio autor

Funkcija radi instancu glavnog `tkinter` sučelja i odmah ga skriva kako bi se izbjeglo prikazivanje praznog sučelja. Zatim poziva metodu za otvaranje sučelja za odabir slike s korisnikovog računala, pri čemu je metodi dodan argument koji ograničava vrstu datoteke koja može biti učitana. Rezultat koji funkcija vraća je cijeli put do odabranog dokumenta.

Slika 6 prikazuje implementaciju funkcije za dodavanje zrnatog šuma na sliku.

```python
def add_speckle_noise(image, var, mean):
    """
    Blagi šum: var=0.01
    Umjereni šum: var=0.1
    Visoki šum: var=0.3
    Vrlo visoki šum: var=0.5

    'speckle'
            Multiplicative noise using ``out = image + n * image``, where
``n`` is Gaussian noise with specified mean & variance.
    """
    image = image / 255.0
    noisy_image = random_noise(image, mode='speckle', var=var, mean=mean)
    noisy_image = (255 * noisy_image).astype(np.uint8)

    return noisy_image
```

Slika 6 - Funkcija za dodavanje zrnatog šuma na sliku
Izvor: Izradio autor

Funkcija kao argument prima nekoliko parametara: sliku, varijancu i srednju vrijednost šuma. U višerednom komentaru napisane su vrijednosti šuma koje se mogu dodati slici i na koji način se zrnati šum generira. Prvi korak funkcije prije dodavanja šuma je da normalizira sliku, odnosno da vrijednosti piksela slike budu u rasponu od [0,1], a ne [0,255]. Razlog tomu je što funkcije za dodavanje šuma obično rade na normaliziranim slikama. Sljedeći korak je dodavanje zrnatog šuma na normaliziranu sliku, a to je postignuto funkcijom `random_noise` iz biblioteke skimage. Funkciji smo predali potrebne argumente koji su:

- Slika
- Tip šuma koji želimo (`mode = „speckle“`)
- Varijancu šuma
- Srednju vrijednost šuma

Zrnati šum se na sliku dodaje tako da se na izvornu sliku doda Gaussov šum i pomnoži s pikselima izvorne slike koji predstavljaju multiplikativnu konstantu. Nakon dodavanja zrnatog šuma na sliku, sliku je potrebno denormalizirati. Slika se denormalizira u

raspon piksela [0,255] i pretvara u tip `uint8`, koji je standardni tip slike. Na kraju funkcija vraća sliku s dodanim zrnatim šumom.

Slika 7 prikazuje implementaciju funkcije za metodu prostornog usrednjavanja.

```
def mean_filter(image):
    kernel_size = 3
    # Dodavanje rubova nula oko slike za rukovanje pikselima na rubu
    padded_image = np.pad(image, ((kernel_size // 2, kernel_size // 2),
(kernel_size // 2, kernel_size // 2)), 'constant', constant_values=0)
    # Inicijalizacija prazne slike za filtriranu sliku
    filtered_image = np.zeros_like(image)

    # Iteracija kroz sve piksele u slici
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            # Izdvajanje područja interesa
            region = padded_image[i:i+kernel_size, j:j+kernel_size]
            # Izračunavanje srednje vrijednosti područja
            filtered_image[i, j] = np.mean(region)

    return filtered_image
```

Slika 7 - Funkcija metode prostornog usrednjavanja
Izvor: Izradio autor

Funkcija kao argument prima sliku na kojoj želim iskoristiti metodu. Prvo postavljamo veličinu kernela koja će se koristiti, a zatim koristimo funkciju `np.pad` koja dodaje rubove oko slike. U funkciji `np.pad` definiramo koliko će se vrijednosti (u ovom slučaju 0) postaviti oko slike. U slučaju kernela 3x3 dodaje se jedan redak i jedan stupac sa svake strane. Nakon dodavanja rubova stvara se nova slika istih dimenzija kao ulazna slika, ali sa svim vrijednostima 0. Ova nova slika će sadržavati filtrirane vrijednosti piksela. Zatim slijedi glavni dio funkcije u kojemu se pomoću dvostruke petlje prolazi kroz svaki piksel na slici i izdvaja 3x3 područje koje je centrirano oko odabranog piksela (i, j). To područje se zatim koristi za izračunavanje srednje vrijednosti i ta se srednja vrijednost dodaje trenutnom pikselu u novokreiranoj slici sa svim nulama. Na kraju funkcija vraća novu sliku koja sadrži nove vrijednosti piksela.

Slika 8 prikazuje implementaciju funkcija za izračun metrike PSNR i metrike SSIM.

```python
# Funkcija za izračunavanje PSNR
def calculate_psnr(img1, img2):
    return psnr(img1, img2, data_range=img1.max() - img1.min())

# Funkcija za izračunavanje SSIM
def calculate_ssim(img1, img2):
    s, _ = ssim(img1, img2, full=True)
    return s
```

Slika 8 - Funkcije metrika PSNR i SSIM
Izvor: Izradio autor

Nakon što je šum uklonjen korištenjem metode potrebno je izračunati metrike PSNR i SSIM. Obje funkcije kao argument primaju dvije slike, prva slika je ulazna slika, a druga je ona koju želimo usporediti s ulaznom. Prva funkcija za računanje PSNR-a vraća vrijednost koju daje funkcija za računanje PSNR-a koju smo uvezli iz biblioteke `skimage.metrics`. Ta funkcija koju smo uvezli iz biblioteke kao argument prima dvije slike i raspon podataka slike. Raspon je potreban jer slike mogu imati različite raspone piksela, a zato što smo raspon definirali pomoću ulazne slike znamo da je raspon dobar. Druga funkcija koja je implementirana izračunava SSIM i koristi funkciju iz biblioteke `skimage.metrics`. Ta ugrađena funkcija vraća SSIM vrijednost, kao i dodatne informacije. Rezultat se sprema u varijablu s i ta varijabla se vraća kao rezultat metrike.

Slika 9 prikazuje glavnu funkciju implementacije metode prostornog usrednjavanja.

```python
def main():
    # Odabir slike
    file_path = choose_image()
    if not file_path:
        print("Nijedna datoteka nije odabrana. Izlazim...")
        return

    # Učitavanje slike u sivoj skali
    image = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
    if image is None:
        print("Učitavanje slike nije uspjelo. Izlazim...")
        return
```

```
    # Dodavanje speckle šuma na sliku
    noisy_image = add_speckle_noise(image, var=0.2, mean=0.0)

    start_time = time.time()

    # Primjena metode prostornog usrednjavanja na sliku sa šumom
    denoised_image = mean_filter(noisy_image)

    end_time = time.time()
    processing_time = end_time - start_time

    # Izračunavanje PSNR i SSIM između originalne slike i slike sa šumom
    psnr_noisy = calculate_psnr(image, noisy_image)
    ssim_noisy = calculate_ssim(image, noisy_image)

    # Izračunavanje PSNR i SSIM između originalne slike i slike nakon
uklanjanja šuma
    psnr_denoised = calculate_psnr(image, denoised_image)
    ssim_denoised = calculate_ssim(image, denoised_image)

    print(f'PSNR između originalne slike i slike sa šumom: {psnr_noisy}
dB')
    print(f'SSIM između originalne slike i slike sa šumom: {ssim_noisy}')
    print(f'PSNR između originalne slike i slike nakon uklanjanja šuma:
{psnr_denoised} dB')
    print(f'SSIM između originalne slike i slike nakon uklanjanja šuma:
{ssim_denoised}')
    print(f'Vrijeme potrebno za uklanjanje šuma metodom prostornog
usrednjavanja: {processing_time:.4f} sekundi')

    # Prikazivanje originalne slike, slike sa šumom i slike nakon
uklanjanja šuma
    plt.figure(figsize=(25, 5))

    plt.subplot(1, 4, 1)
```

```
    plt.title('Originalna slika')
    plt.imshow(image, cmap='gray')
    plt.axis('off')

    plt.subplot(1, 4, 2)
    plt.title('Slika sa šumom\nPSNR: {:.2f} dB\nSSIM:
{:.4f}'.format(psnr_noisy, ssim_noisy))
    plt.imshow(noisy_image, cmap='gray')
    plt.axis('off')

    plt.subplot(1, 4, 3)
    plt.title('Slika nakon uklanjanja šuma\nPSNR: {:.2f} dB\nSSIM:
{:.4f}'.format(psnr_denoised, ssim_denoised))
    plt.imshow(denoised_image, cmap='gray')
    plt.axis('off')

    plt.subplot(1, 4, 4)
    plt.axis('off')
    plt.text(0.5, 0.5, f'Vrijeme izvođenja:\n{processing_time:.4f}
sekundi',
             horizontalalignment='center', verticalalignment='center',
             fontsize=14, color='black', transform=plt.gca().transAxes)
    plt.show()

if __name__ == "__main__":
    main()
```

Slika 9- Glavna funkcija metode prostornog usrednjavanja
Izvor: Izradio autor

Zadnji korak je definirati funkciju koja će sve korake spojiti. Ova funkcija upravlja i poziva sve ostale funkcije kako bi dobila i prikazala konačne rezultate. Prvo se poziva funkcija za učitavanje putanje do slike. Ako putanja do slike postoji funkcija nastavlja s izvođenjem, a ako putanja ne postoji funkcija ispisuje prikladnu poruku. Ako funkcija nastavlja s izvođenjem sljedeći korak ja da se slika učitava u sivoj skali i zatim se provjerava je li slika učitana. Nakon učitavanja slike u sivoj skali, poziva se funkcija za dodavanje zrnatog šuma na sliku i ta slika joj se predaje kao argument zajedno s vrijednostima varijance i srednje vrijednosti šuma. Nakon što je dodan zrnati šum na

sliku, poziva se funkcija za uklanjanje šuma sa slike kojoj se kao argument predaje slika s dodanim šumom. Nakon izvršavanja uklanjanja šuma pozivaju se funkcije za izračun PSNR-a i SSIM-a kojima se kao argumenti predaju ulazna(originalna) slika i slika s uklonjenim šumom. Zatim se dobiveni rezultati ispisuju i u novom sučelju se prikazuju sve tri slike (ulazna, s dodanim šumom i s uklonjenim šumom) zajedno s odgovarajućim vrijednostima PSNR-a i SSIM-a.

6. Metoda korištenja valića

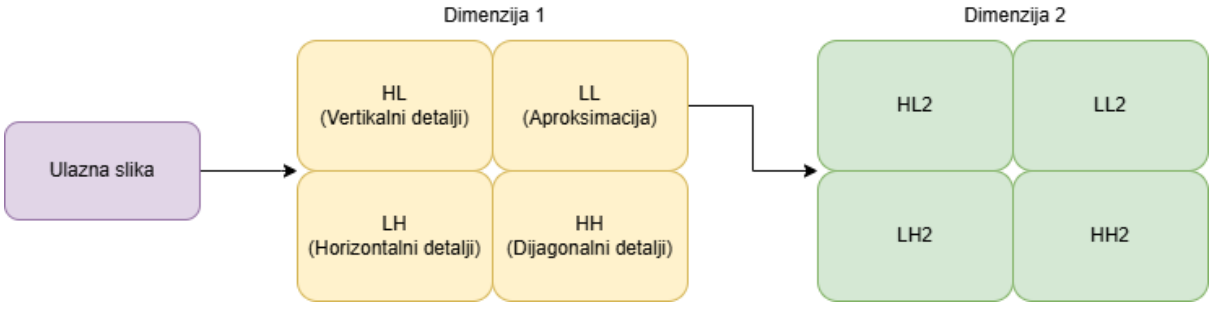
Metoda korištenja valića moćna je tehnika u obradi slike za uklanjanje šuma, koja omogućava očuvanje bitnih detalja slike. U usporedbi sa standardnim metodama koje rade u frekvencijskoj domeni (primjerice Fourierova transformacija), metoda korištenja valića kombinira frekvencijsku i vremensku analizu (Bnou, et al., 2020). Metoda korištenja valića može se koristiti za:

- Uklanjanje šuma
- Kompresiju slike
- Rekonstrukciju slike
- Isticanje značajki na slici (Bnou, et al., 2020)

Koraci kojima se uklanja šum pomoću ove metode su:

1. Valićna transformacija
2. Binarno ograničavanje
3. Inverzna valićna transformacija (Bnou, et al., 2020)

Za valićnu transformaciju najčešće se koristi diskretna valićna transformacija (*engl. Discrete Wavelet Transform, DWT*), a osim nje postoji ih još nekoliko. Kada se koristi diskretna valićna transformacija na 2D signalima kao što su slike, ona sliku razdvaja na četiri potpodručja. Ta potpodručja su aproksimacija (LL), horizontalni detalji (LH), vertikalni detalji (HL) i dijagonalni detalji (HH), što se može vidjeti na Slici 10. Razlog zašto se to događa je jer se primjenjuje formula za diskretnu valićnu transformaciju i na stupcima i na retcima slike (Bnou, et al., 2020).



Slika 10 - Prikaz primjene DWT-a na sliku
Izvor: Izradio autor

Formulom za 1D diskretnu valićnu transformaciju, analogni signal $x(t)$ s konačnom energijom možemo razdvojiti na zbroj pomaknutih i proširenih valićnih funkcija $\Psi(t)$ i pomaknutih funkcija skale $\varphi(t)$ (Bnou, et al., 2020):

$$x(t) = \sum_{k=-\infty}^{\infty} c(k)\phi(t-k) + \sum_{j=0}^{\infty}\sum_{k=-\infty}^{\infty} d(j,k)2^{\frac{j}{2}}\psi\left(2^j t - k\right) \tag{18}$$

gdje su: $c(k)$ aproksimacijski koeficijent (predstavlja niskofrekvencijski dio signala), $d(j,k)$ detaljni koeficijenti (predstavlja visokofrekvencijske dijelove signala), $\phi(t-k)$ skalirajuća funkcija (predstavlja bazne funkcije za aproksimaciju, niskopropusni filter), $2^{\frac{j}{2}}$ normalizacijski faktor (osigurava očuvanje energije kroz skale) i $\psi(2^j t - k)$ valićna funkcija (predstavlja funkcije za detalje, uzima varijacije na različitim skalama j i pozicijama k). Prvi dio izraza (dio prije +) zbraja skalirane i pomaknute verzije skalirajuće funkcije kako bi se predstavile niskofrekvencijske komponente, dok drugi dio formule (dio nakon +) zbraja skalirane i pomaknute verzije valićne funkcije na različitim skalama i pozicijama radi predstavljanja visokofrekvencijskih komponenti. Formula za aproksimacijski koeficijent glasi (Bnou, et al., 2020):

$$c(k) = \int_{-\infty}^{+\infty} x(t)\phi(t-k)dt \tag{19}$$

Formula za izračun detaljnih koeficijenata je (Bnou, et al., 2020):

$$d(j,k) = 2^{\frac{j}{2}}\int_{-\infty}^{+\infty} x(t)\psi\left(2^j t - l\right)dt \tag{20}$$

Glavnu formulu za diskretnu valićnu transformaciju na 1D signalima možemo pretvoriti u formulu za 2D signale, odnosno slike. Ako želimo napraviti diskretnu valićnu transformaciju, onda se signal prikazuje kao slika $I(x,y)$ gdje su x i y kordinate na slici gdje se primjenjuje valićna transformacija, a formula izgleda ovako (Bnou, et al., 2020):

$$\begin{aligned} I(x,y) = {} & \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} c(m,n)\,\phi(x-m,y-n) + \\ & \sum_{j=0}^{\infty}\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} \begin{matrix} d_H(j,m,n)\psi_H\left(2^j x - m, y - n\right) + \\ d_V(j,m,n)\psi_H\left(x - m, 2^j y - n\right) + \\ d_D(j,m,n)\psi_D\left(2^j x - m, 2^j y - n\right) \end{matrix} \end{aligned} \tag{21}$$

Razlika između formule za 1D i 2D signale je što za 2D signale u formuli se koriste osim koordinata na slici i indeksi za skalirajuće i valićne funckije (horizontalno m, vertikalno n, predstavlja pomake primijenjene na skalirajuću funkciju i valićne funkcije) i indeks skale (j, predstavlja razinu valićne transfrmacije, veće vrijednosti su komponente viših frekvencija, a niže vrijednosti komponente nižih frekvencija). Također u formuli za 2D detaljni koeficijenti na skali i valićne funkcije na skali računaju se za horizontalne, vertikalne i dijagonalne komponente. Također, osim glavne formule, formula za aproksimacijski koeficijenti ($c(m,n)$) i formula za izračun detaljnih koeficjenata ($d(j,m,n)$) mogu se proširiti kako bi se koristile na 2D signalima (Bnou, et al., 2020). Tada one izgledaju ovako:

Formula za aproksimacijski koeficijent (Bnou, et al., 2020):

$$c(m,n) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(x,y)\phi(x-m, y-n)dxdy \tag{22}$$

Formule za izračun detaljnih koeficijenata (Bnou, et al., 2020):

horizontalno:

$$d_H(j,m,n) = 2^{\frac{j}{2}} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(x,y)\psi_H\left(2^j x - m, y - n\right) dxdy \tag{23}$$

vertikalno:

$$d_V(j,m,n) = 2^{\frac{j}{2}} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(x,y)\psi_V\left(x - m, 2^j y - n\right) dxdy \tag{24}$$

dijagonalno:

$$d_D(j,m,n) = 2^{\frac{j}{2}} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(x,y)\psi_D\left(2^j x - m, 2^j y - n\right) dxdy \tag{25}$$

Nakon što je odrađen prvi korak i napravljena je valićna transformacija, potrebno je napraviti binarno ograničavanje (*engl. thresholding*). Ovaj korak je bitan jer koeficijenti koji su uzrokovani šumom (koeficijenti ispod praga) postavljaju se na vrijednost 0, a oni koji nisu postavljaju se na vrijednost 1. Postoje dva načina dodavanja praga, tvrdo binarno ograničavanje (*engl. hard thresholding*) i meko binarno ograničavanje (*engl. soft thresholding*) (Lopac, 2022).

Formula tvrdog binarnog ograničavanja izgleda ovako (Bnou, et al., 2020):

$$\mathrm{d}'_{H,V,D}(j,m,n) = \begin{cases} T\ ako\ je\ \left|d_{H,V,D}(j,m,n)\right| > T \\ 0\ ako\ je\ \left|d_{H,V,D}(j,m,n)\right| \leq T \end{cases} \quad (26)$$

gdje su $\mathrm{d}'_{H,V,D}(j,m,n)$ detaljni koeficijent dobiveni nakon postupka binarnog ograničavanja, $d_{H,V,D}(j,m,n)$ originalni koeficijenti i T je vrijednost praga. Ako je originalni koeficijent veći od praga, on ostaje nepromijenjen, a ako je manji, postavlja se na vrijednost 0 (Bnou, et al., 2020).

Za meko binarno ograničavanje formula izgleda ovako (Bnou, et al., 2020):

$$\mathrm{d}'_{H,V,D}(j,m,n) = sign\left(d_{H,V,D}(j,m,n)\right) \cdot \max\left(\left|d_{H,V,D}(\mathrm{j,m,n}) - \mathrm{T}, 0\right|\right) \quad (27)$$

$sign$ funkcija vraća predznak +1 ako je ono što je u zagradi pozitivno i –1 ako je negativno. Ako je rezultat max funkcije veći od 0, on se množi s predznakom koeficijenta, a ako je manja ili jednaka 0 rezultat postaje 0 (Bnou, et al., 2020).

Nakon što je odrađeno binarno ograničavanje, potrebno je napraviti inverznu diskretnu valićnu transformaciju (*engl. Inverse Discrete Wavelet Transform, IDWT*). IDWT dobijemo tako da u formulu za računanje diskretne valićne transformacije ubacimo novo dobivene vrijednosti nakon binarnog ograničavanja (Bnou, et al., 2020).

6.1. Implementacija

Za implementaciju metode korišenja valića iskorištene su neke funkcije koje su napravljene za prijašnju metodu kao što su funkcije za dodavanje šuma na sliku i funkcije za izračun metrika kvalitete PSNR i SSIM. Osim toga, glavna funkcija koda je ostala većim djelom ista, osim što je umjesto korištenja funkcije za računanje metode prostornog usrednjavanja korištena nova funkcija koja radi na principu metode korištenja valića. Nadalje će biti objašnjeni dijelovi koda koji su drugačiji od onih u prijašnjoj metodi. Kako je cijela metoda korištenja valića podijeljena na tri dijela, tako su i u kodu izdvojene funkcije za pojedini dio i onda pozivane određenim redom kako bi se dobila funkcionalnost metode. Prvo su definirane funkcije za DWT i inverznu IDWT, a nakon toga funkcija za binarno ograničavanje.

Slika 11 prikazuje funkciju za DWT.

```python
def dwt2(image, target_levels):
    h, w = image.shape
    if h % 2 != 0:
        image = np.pad(image, ((0, 1), (0, 0)), 'constant')
        h += 1
    if w % 2 != 0:
        image = np.pad(image, ((0, 0), (0, 1)), 'constant')
        w += 1

    LL = (image[0::2, 0::2] + image[1::2, 0::2] + image[0::2, 1::2] +
image[1::2, 1::2]) / 4
    LH = (image[0::2, 0::2] - image[1::2, 0::2] + image[0::2, 1::2] -
image[1::2, 1::2]) / 4
    HL = (image[0::2, 0::2] + image[1::2, 0::2] - image[0::2, 1::2] -
image[1::2, 1::2]) / 4
    HH = (image[0::2, 0::2] - image[1::2, 0::2] - image[0::2, 1::2] +
image[1::2, 1::2]) / 4

    if target_levels == 1:
        return LL, [(LH, HL, HH)]

    LL, sub_details = dwt2(LL, target_levels - 1)
    return LL, sub_details + [(LH, HL, HH)]
```

Slika 11 - Funkcija diskretne valićne transformacije
Izvor: Izradio autor

Prvi korak implementacije je napraviti funkciju koja izračunava diskretnu valićnu transformaciju na 2D signalu uz mogućnost rada na više razina (dimenzija). Princip rada na više razina prikazan je na prethodno objašnjenoj Slici 10. Argumenti funkcije su slika (2D `NumPy` polje) i broj razina koje metoda treba izvesti. Prvi korak kod funkcije je da se osigura da dimenzije slike budu parne; ako je jedna dimenzija (redak ili stupac) neparna, dodaje se određena dimenzija. Ta funkcionalnost napravljena je pomoću `np.pad` (padding) metode, koja dodaje redak na dno ili stupac na desnu stranu ako je potrebno, pri čemu je vrijednost koja se dodaje 0. Zatim se slika dijeli na potpodručja LL (aproksimacija), LH (horizontalni detalji), HL (vertikalni detalji) i HH (dijagonalni

detalji). Svaka pojedina komponenta dobivena je tako da se uzima svaki drugi redak ili stupac i da se počinje od nultog ili prvog. Razlog za to je kako bi se simuliralo smanjivanje rezolucije (*engl. downsampling*). Nadalje se provjerava je li DWT odrađen na onoliko razina koliko je to postavljeno, a ako je broj razina jednak 1, onda to znači da je odrađeno na svim zadanim razinama. Ako broj razina nije jednak 1, onda funkcija poziva samu sebe (rekurzivno), ali umjesto ulazne slike radi na aproksimaciji koja je dobivena iz prvog izvođenja. Funkcija vraća zadnje vrijednosti LL, LH, HL, HH koje su izračunate za određenu razinu.

Slika 12 prikazuje funkciju za IDWT.

```
def idwt2(LL, details):
    if not details:
        return LL

    LH, HL, HH = details[0]
    h, w = LL.shape
    h_lh, w_lh = LH.shape
    h_hl, w_hl = HL.shape
    h_hh, w_hh = HH.shape

    if (h, w) != (h_lh, w_lh):
        LH = cv2.resize(LH, (w, h))
    if (h, w) != (h_hl, w_hl):
        HL = cv2.resize(HL, (w, h))
    if (h, w) != (h_hh, w_hh):
        HH = cv2.resize(HH, (w, h))

    image = np.zeros((h * 2, w * 2), dtype=np.float64)
    image[0::2, 0::2] = LL + LH + HL + HH
    image[1::2, 0::2] = LL - LH + HL - HH
    image[0::2, 1::2] = LL + LH - HL - HH
    image[1::2, 1::2] = LL - LH - HL + HH
    return idwt2(image, details[1:])
```

Slika 12 - Funkcija inverzne diskretne valične transformacije
Izvor: Izradio autor

Kako bi smo sliku vratili i prvobitni oblik nakon DWT-a, potrebno je napraviti IDWT. Parametri koje funkcija prima su aproksimacija (LL) i detalji odnosno sve ostale

vrijednosti potpodručja (HL, LH, HH). Prvo se provjeravaju detalji i ako je to prazna lista funkcija vraća LL jer tada više nema razina koje se trebaju obraditi. Nakon provjere detalja provjeravaju se dimenzije slike i ako su dimenzije slike neparne, slici se mijenjaju dimenzije. Sljedeći korak je da se iz liste detalja dobiju LH, HL, HH za trenutnu razinu i da se iz LL (aproksimacije) izvuku trenutne dimenzije slike. Nakon što su dobivene dimenzije, stvara se prazna slika koja je dvostruko većih dimenzija; razlog je što se ovim procesom vraća na veću razinu pa je i slika veća. Zatim se prazna slika popunjava vrijednostima LL, LH, HL i HH u svakom drugom retku i svakom drugom stupcu, počevši od nultog ili prvog. Funkcija poziva samu sebe koristeći trenutnu sliku kao novi LL i preostale detalje (ostale vrijednosti za LH, HL, HH).

Slika 13 prikazuje funkcije za primjenu mekog i tvrdog binarnog ograničavanja.

```
# Funkcija za soft thresholding
def soft_thresholding(data, threshold):
    return np.sign(data) * np.maximum(np.abs(data) - threshold, 0)

# Funkcija za hard thresholding
def hard_thresholding(data, threshold):
    return data * (np.abs(data) > threshold)
```

Slika 13 - Funkcije za binarno ograničavanje
Izvor: Izradio autor

Koraci metode zahtijevaju binarno ograničavanje. Prva funkcija je implementacija mekog binarnog ograničavanja, a druga funkcija implementacija tvrdog binarnog ograničavanja. Prva funkcija (meko binarno ograničavanje) vraća ili 0 ili vrijednosti koje su veće od praga umanjene za vrijednost praga. Pomoću `np.maximum` se osigurava da vrijednosti manje od praga i manje od 0 budud postavljene na vrijednost 0. Točnije ako apsolutne vrijednosti umanjene za prag budu pozitivne, `np.maximum` ih ostavlja neizmijenjene, a ako su negativne, postavljaju se u 0. Funkcija `np.sign` vraća predznak vrijednosti (+1 ili -1), pri čemu se sve vrijednosti množe s tim rezultatom i dobiva se rezultat mekog binarnog ograničavanja.

Druga funkcija (tvrdo binarno ograničavanje) jednostavnija je metoda binarnog ograničavanja. Provjerava se je li apsolutna vrijednost elementa veća od praga i kada se množi s ulaznim elementom (data), vrijednosti veće od praga ostaju

nepromijenjene, a vrijednosti manje od praga postavljaju se u 0. Rezultat koji se dobije je ono što funkcija vraća.

Slika 14 prikazuje konačnu implementaciju funkcije koja koristi sve ostale potrebne funkcije kako bi se napravila potpuna funkcionalnost metode korištenja valića.

```
def wavelet_filter(image, target_levels, threshold, threshold_type):
    # Konvertiranje slike u float64 radi preciznosti
    image = image.astype(np.float64)

    # Primjena DWT
    LL, details = dwt2(image, target_levels)
    print(details)

    # Primjena odabranog thresholdinga
    if threshold_type == 'soft':
        details = [(soft_thresholding(d[0], threshold),
                    soft_thresholding(d[1], threshold),
                    soft_thresholding(d[2], threshold)) for d in details]
    elif threshold_type == 'hard':
        details = [(hard_thresholding(d[0], threshold),
                    hard_thresholding(d[1], threshold),
                    hard_thresholding(d[2], threshold)) for d in details]

    # Primjena IDWT
    denoised_image = idwt2(LL, details)

    # Vraćanje slike na originalne dimenzije ako su promijenjene
    denoised_image = denoised_image[:image.shape[0], :image.shape[1]]

    # Konvertiranje natrag u uint8
    denoised_image = np.clip(denoised_image, 0, 255).astype(np.uint8)
    return denoised_image, LL, details
```

Slika 14 - Funkcija metode korištenja valića
Izvor: Izradio autor

Kao argumente funkcija prima sliku, broj razina na koliko će raditi, prag i vrstu binarnog ograničavanja (meko, tvrdo). Prvi korak je da se ulazna slika pretvara u float64 tip

podatka radi veće preciznosti algoritma. Zatim se poziva funkcija za izračun DWT-a, kojoj se kao argumenti predaju slika i razina do koje se treba izvoditi, a funkcija vraća dva rezultata koji se prikladno spremaju. Rezultati su LL (aproksimacija) i detalji (LH, HL, HH). Zatim, s obzirom na to koji je tip binarnog ograničavanja odabran na detaljima koji su dobiveni, izvodi se funkcija za to binarno ograničavanje. Nakon binarnog ograničavanja slika se vraća u izvorni oblik pomoću funkcije za IDWT. Rezultat funkcije za IDWT provjerava se kako bi se utvrdilo jesu li dimenzije slike promijenjene. Ako su promijenjene, dimenzije se vraćaju na izvorne, a slika se zatim pretvara natrag u `uint8` tip podatka. Kako bi se osiguralo da su svi pikseli u rasponu od 0 do 255 koristi se metoda `np.clip`. Na kraju cijela funkcija metode korištenja valića vraća sliku bez šuma, LL (zadnju aproksimaciju) i zadnje detalje (LH, HL, HH).

Slika 15 prikazuje glavnu funkciju implementacije metode korištenja valića.

```
def main():
    # Odabir slike
    file_path = choose_image()
    if not file_path:
        print("Nijedna datoteka nije odabrana. Izlazim...")
        return

    # Učitavanje slike u grayscale
    image = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
    if image is None:
        print("Učitavanje slike nije uspjelo. Izlazim...")
        return

    # Dodavanje zrnatog šuma slici
    noisy_image = add_speckle_noise(image, var=0.2, mean=0.0)

    start_time = time.time()

    # Primjena wavelet filtriranja na sliku sa šumom
    target_levels = 1  # Broj razina za DWT
```

```
    denoised_image_soft, LL_soft, details_soft =
wavelet_filter(noisy_image, target_levels, threshold=25,
threshold_type='soft')
    denoised_image_hard, LL_hard, details_hard =
wavelet_filter(noisy_image, target_levels, threshold=25,
threshold_type='hard')

    # Provjera jesu li dimenzije slike jednake
    if denoised_image_soft.shape != image.shape or
denoised_image_hard.shape != image.shape:
        print("Dimenzije filtriranih slika se ne podudaraju s originalom.
Izlazim...")
        return

    end_time = time.time()
    processing_time = end_time - start_time

    # Izračun PSNR i SSIM između originalne slike i slike sa šumom
    psnr_noisy = calculate_psnr(image, noisy_image)
    ssim_noisy = calculate_ssim(image, noisy_image)

    # Izračun PSNR i SSIM između originalne slike i slike nakon uklanjanja
šuma
    psnr_denoised_soft = calculate_psnr(image, denoised_image_soft)
    ssim_denoised_soft = calculate_ssim(image, denoised_image_soft)
    psnr_denoised_hard = calculate_psnr(image, denoised_image_hard)
    ssim_denoised_hard = calculate_ssim(image, denoised_image_hard)

    print(f'PSNR između originalne i slike sa šumom: {psnr_noisy} dB')
    print(f'SSIM između originalne i slike sa šumom: {ssim_noisy}')
    print(f'PSNR između originalne i slike nakon uklanjanja šuma(soft
thresholding): {psnr_denoised_soft} dB')
    print(f'SSIM između originalne i slike nakon uklanjanja šuma(soft
thresholding): {ssim_denoised_soft}')
    print(f'PSNR između originalne i slike nakon uklanjanja šuma(hard
thresholding): {psnr_denoised_hard} dB')
```

```
    print(f'SSIM između originalne i slike nakon uklanjanja šuma(hard
thresholding): {ssim_denoised_hard}')
    print(f'Vrijeme potrebno za uklanjanje šuma metodom korištenja valića:
{processing_time:.4f} sekundi')

    # Prikaz rezultata
    plt.figure(figsize=(25, 15))

    plt.subplot(3, 5, 1)
    plt.title('Originalna slika')
    plt.imshow(image, cmap='gray')
    plt.axis('off')

    plt.subplot(3, 5, 2)
    plt.title('Slika sa šumom\nPSNR: {:.2f} dB\nSSIM:
{:.4f}'.format(psnr_noisy, ssim_noisy))
    plt.imshow(noisy_image, cmap='gray')
    plt.axis('off')

    plt.subplot(3, 5, 3)
    plt.title('Slika nakon uklanjanja šuma (Soft)\nPSNR: {:.2f} dB\nSSIM:
{:.4f}'.format(psnr_denoised_soft, ssim_denoised_soft))
    plt.imshow(denoised_image_soft, cmap='gray')
    plt.axis('off')

    plt.subplot(3, 5, 4)
    plt.title('Slika nakon uklanjanja šuma (Hard)\nPSNR: {:.2f} dB\nSSIM:
{:.4f}'.format(psnr_denoised_hard, ssim_denoised_hard))
    plt.imshow(denoised_image_hard, cmap='gray')
    plt.axis('off')

    plt.subplot(3, 5, 5)
    plt.title('LL')
    plt.imshow(LL_soft, cmap='gray')
    plt.axis('off')
```

```
    # Prikaz LH, HL, HH potpodručja za soft thresholding
    LH_soft, HL_soft, HH_soft = details_soft[0]
    plt.subplot(3, 5, 6)
    plt.title('LH (Soft)')
    plt.imshow(LH_soft, cmap='gray')
    plt.axis('off')

    plt.subplot(3, 5, 7)
    plt.title('HL (Soft)')
    plt.imshow(HL_soft, cmap='gray')
    plt.axis('off')

    plt.subplot(3, 5, 8)
    plt.title('HH (Soft)')
    plt.imshow(HH_soft, cmap='gray')
    plt.axis('off')

    # Prikaz LH, HL, HH potpodručja za hard thresholding
    LH_hard, HL_hard, HH_hard = details_hard[0]
    plt.subplot(3, 5, 9)
    plt.title('LH (Hard)')
    plt.imshow(LH_hard, cmap='gray')
    plt.axis('off')

    plt.subplot(3, 5, 10)
    plt.title('HL (Hard)')
    plt.imshow(HL_hard, cmap='gray')
    plt.axis('off')

    plt.subplot(3, 5, 11)
    plt.title('HH (Hard)')
    plt.imshow(HH_hard, cmap='gray')
    plt.axis('off')

    plt.subplot(3, 5, 12)
    plt.axis('off')
```

```
    plt.text(0.5, 0.5, f'Vrijeme izvođenja:\n{processing_time:.4f}
sekundi',
             horizontalalignment='center', verticalalignment='center',
             fontsize=14, color='black', transform=plt.gca().transAxes)

    plt.show()

if __name__ == "__main__":
    main()
```

Slika 15 - Glavna funkcija metode korištenja valića
Izvor: Izradio autor

Kao i na prijašnjoj metodi u glavnoj funkciji učitava se slika i rade se provjere. Nakon što su prošle provjere poziva se funkcija za odrađivanje metode korištenja valića, koja je prethodno objašnjena. Zatim se računaju metrike između slike bez šuma i slike s dodanim šumom, a zatim slike bez šuma i slike nakon izvođenja metode. Metoda se izvodi sa oba načina binarnog ograničavanja i metrike se računaju s rezultatima oba načina. Nakon izračunavanja metrika prikazuju se ulazna slika, slika sa šumom i slike nakon izvođenja metode. Osim toga prikazuje se i aproksimacija (LL) i svi ostali detalji (HL, LH, HH) za krajnju razinu.

7. Metoda nelokalnog usrednjavanja

Za razliku od metoda lokalnog usrednjavanja, kao što je prva obrađena metoda, koje uzimaju srednju vrijednost piksela koji okružuju odabrani piksel kako bi uklonile šum, metoda nelokalnog usrednjavanja uzima srednju vrijednost svih piksela na slici, ponderirano tome koliko su pikseli slični ciljanom pikselu. Dakle, metoda se temelji na pretpostavci da na cijeloj slici postoje slični pikseli, pa čak i ako nisu susjedni. Glavna ideja metode nelokalnog usrednjavanje je da se može predstaviti piksel s ponderiranim prosjekom svih sličnih piksela na slici. Za razliku od metode lokanlog usrednjavanja koja uzima jedno susjedstvo, metoda nelokalnog usrednjavanja uspoređuje susjedstvo oko svakog piksela sa susjedstvima oko drugih piksela (Buades, et al., 2011). Način kako metoda nelokalnog usrednjavanja dolazi do rješenja je sljedeći:

1. Izvlačenje susjedstva
2. Usporedba susjedstva
3. Izračunavanje težina
4. Ponderirani prosjek (Buades, et al., 2011)

Kroz ova četiri koraka metoda uklanja šum sa slike. U prvom koraku (izvlačenje susjedstva) izvlači se susjedstvo $B(p,f)$ oko piksela p veličine (Buades, et al., 2011):

$$(2f+1)\times(2f+1) \tag{28}$$

pri čemu f predstavlja broj piksela oko odabranog piksela. Dakle, ako je $f=1$, susjedstvo će biti veličine 3x3. Drugi korak (usporedba susjedstva) izračunava kvadrat euklidske udaljenosti između dva susjedstva, onog susjedstva na trenutnom pikselu $B(p,f)$ i drugog susjedstva $B(q,f)$. Kvadrat euklidske udaljenosti između susjedstva predstavlja mjeru sličnosti između dva susjedstva, a računa se kao (Buades, et al., 2011):

$$d^2\big(B(p,f),B(q,f)\big)=\frac{1}{(2f+1)^2}\sum_{j\epsilon B(0,f)}\big(u(p+j)-u(q+j)\big)^2 \tag{29}$$

gdje $\frac{1}{(2f+1)^2}$ predstavlja normalizacijski faktor, a $(u(p+j)-u(q+j))^2$ je razlika u intenzitetu između piksela.

Naravno, metoda se može primijeniti i na slikama u boji, odnosno RGB (*engl. Red, Green, Blue*) slikama. Ako se formula koristi na RGB slici, tada ona izgleda ovako (Buades, et al., 2011):

$$d^2\big(B(p,f),B(q,f)\big)=\frac{1}{3(2f+1)^2}\sum_{i=1}^{3}\sum_{j\epsilon B(0,f)}\big(u_i(p+j)-u_i(q+j)\big)^2 \qquad (30)$$

U formuli za RGB sliku u_i predstavlja intenzitet boje za sliku u kanalu i. Također, u ovoj verziji formule ona iterira kroz sva tri kanala boja za RGB sliku (crveni, zeleni, plavi) (Buades, et al., 2011).

Kod trećeg koraka (izračunavanje težine) na temelju kvadrata euklidske udaljenosti izračunava se težina $w(p,q)$ korištenjem eksponencijalne funkcije (Buades, et al., 2011):

$$w(p,q)=\exp\left(-\frac{\max d^2\big(B(p,f),B(q,f)\big)-2\,\sigma^2,0.0}{h^2}\right) \qquad (31)$$

gdje je σ standardna devijacija šuma, odnosno prag ispod kojeg razlike između susjedstva mogu biti uzrokovane šumom, a h je parametar filtriranja. Veće vrijednosti h daju ravnomjernije ponderiranje sličnih susjedstva, dok manje vrijednosti stavljaju naglasak na najbliža susjedstva. Vrijednost za h obično se postavlja prema σ, $h=k\sigma$, gdje je k konstanta (Buades, et al., 2011).

Formula za izračun težine može se promijeniti kako bi se koristila na RGB slici, razlika je samo u tome što se koristi formula kvadratne euklidske udaljenosti za RGB sliku.

$$w(p,q)=\exp\left(-\frac{\max\frac{1}{3(2f+1)^2}\sum_{i=1}^{3}\sum_{j\epsilon B(0,f)}\big(u_i(p+j)-u_i(q+j)\big)^2-2\,\sigma^2,0.0}{h^2}\right) \qquad (32)$$

U četvrtom koraku (ponderirani prosjek) računa se vrijednost piksela kao ponderirani prosjek intenziteta piksela u susjedstvima (Buades, et al., 2011):

$$\hat{u}(p)=\frac{1}{C(p)}\sum_{q\epsilon B(p,r)}u(q)w(p,q) \qquad (33)$$

gdje je $u(q)$ intenzitet piksela u ulaznoj slici. Ova formula se također može koristit za RGB sliku, ali ona onda izgleda ovako (Buades, et al., 2011):

$$\hat{u}_i(p) = \frac{1}{C(p)} \sum_{q \epsilon B(p,r)} u_i(q) w(p,q) \tag{34}$$

Nakon što su svi koraci odrađeni, stvara se nova slika s uklonjenim šumom. Vrijednosti za σ, h, veličinu susjedstva i veličinu bloka za preostale dijelove slike koje metoda koristi prikazane su u Tablici 1 za sliku u sivim tonovima i u Tablici 2 za RGB sliku.

σ	Veličina susjedstva	Veličina bloka	h
$0 < \sigma \leq 15$	3x3	21x21	0.40σ
$15 < \sigma \leq 30$	5x5	21x21	0.40σ
$30 < \sigma \leq 45$	7x7	35x35	0.35σ
$45 < \sigma \leq 75$	9x9	35x35	0.35σ
$75 < \sigma \leq 100$	11x11	35x35	0.30σ

Tablica 1 - Vrijednosti koje metoda nelokalnog usrednjavanja koristi za sliku u sivim tonovima
Izvor: (Buades, et al., 2011)

σ	Veličina susjedstva	Veličina bloka	h
$0 < \sigma \leq 25$	3x3	21x21	0.40σ
$25 < \sigma \leq 55$	5x5	21x21	0.40σ
$55 < \sigma \leq 100$	7x7	35x35	0.35σ

Tablica 2 - Vrijednosti koje metoda nelokalnog usrednjavanja koristi za RGB sliku
Izvor: (Buades, et al., 2011)

7.1. Implementacija

Implementacija je slična kao i u dvije prethodne metode. Sve funkcije koje nisu povezane s načinom rada metode ostale su iste, a to su funkcije za učitavanje slike, dodavanje šuma, funkcije za izračun metrika PSNR i SSIM, te većina glavne funkcije koja sve to spaja u jednu cjelinu. Kao i prijašnje metode, tako je i ova implementirana u programskom jeziku Python. Kako je ova metoda slična metodi prostornog usrednjavanja, jedna funkcija je slična, a to je funkcija za izdvajanje susjedstva oko odabranog piksela. Unatoč sličnosti postoji značajna razlika. Kod metode nelokalnog usrednjavanja potrebno je koristiti susjedstvo oko trenutnog piksela, te oko drugih piksela i zatim izračunavati težinu i kvadrat euklidske udaljenosti. Navedene funkcije pozivaju se u glavnoj funkciji metode koja uklanja šum sa slike.

Slika 16 prikazuje funkciju za izdvajanje susjedstva oko piksela.

```python
def extract_neighborhood(image, p, f):
    padded_image = np.pad(image, ((f, f), (f, f)), 'reflect')
    i, j = p
    neighborhood = padded_image[i:i + 2 * f + 1, j:j + 2 * f + 1]
    return neighborhood
```

Slika 16 - Funkcija za izdvajanje susjedstva
Izvor: Izradio autor

Kod metode nelokalnog usrednjavanja potrebno je više puta izvlačiti susjedstvo piksela, pa je iz tog razloga za to napravljena posebna funkcija. Funkcija kao argumente prima tri vrijednosti: sliku, poziciju piksela i broj piksela oko odabranog piksela. Prvi korak funkcije je da dodaje rub oko slike koji reflektira odabrani red i stupac i zatim iz vrijednosti piksela uzimaju se koordinate i,j. Nakon toga funkcija izdvaja kvadratno susjedstvo oko piksela. Izdvajaju se redovi počevši od i do $i + 2 \cdot f$ i također se izdvajaju stupci počevši od j do $j + 2 \cdot f$, Kada se uzme presjek, dobije se željeno susjedstvo oko piksela. Na kraju funkcija vraća izdvojeno susjedstvo oko odabranog piksela.

Slika 17 prikazuje funkciju za usporedbu dva izdvojena susjedstva.

```python
def compute_weight(image, p, q, f, sigma, h):
    B_p_f = extract_neighborhood(image, p, f)
    B_q_f = extract_neighborhood(image, q, f)
    d_squared = np.sum((B_p_f - B_q_f) ** 2) / (2 * f + 1) ** 2
    weight = np.exp(-max(d_squared - 2 * sigma ** 2, 0.0) / h ** 2)
    return weight
```

Slika 17 - Funkcija za izračun težine
Izvor: Izradio autor

Metoda zahtijeva da se dva susjedstva uspoređuju, pa je zato implementirana funkcija za izračun težine i euklidske udaljenosti dva susjedstva. Funkcija kao argumente prima sliku, piksel, drugi piksel, broj piksela oko odabranih piksela, standardnu devijaciju šuma i parametar za prilagodbu težine koji kontrolira glatkoću težine. Prvi korak funkcije je da izdvoji susjedstvo glavnog piksela (p) i susjedstvo piksela s kojim se uspoređuje (q), a da bi se to napravilo poziva se funkcija za izvlačenje susjedstva. Nakon što su izdvojena susjedstva, izračunava se kvadrat euklidske udaljenosti između elemenata susjedstva oko piksela p (B_p_f) i piksela q (B_q_f). Ako su susjedstva jako slična, rezultat će biti mala vrijednost, a ako su različita rezultat će biti

veći. Nakon izračuna kvadrata euklidske udaljenosti računa se težina koristeći kvadratnu euklidsku udaljenost. Ako su susjedstva slična, težina će biti blizu 1, a ako su različita, težina će biti manja (bliže 0). Na kraju funkcija vraća izračunatu težinu.

Slika 18 prikazuje funkciju koja izvodi korake metode nelokalnog usrednjavanja.

```
def nonlocal_means_filter(image, f=1, h=10.0, sigma=10.0):
    filtered_image = np.zeros_like(image, dtype=np.float64)
    padded_image = np.pad(image, ((f, f), (f, f)), 'reflect')
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            p = (i + f, j + f)
            weights = []
            neighborhood_weights = []
            for m in range(-f, f + 1):
                for n in range(-f, f + 1):
                    q = (i + m + f, j + n + f)
                    weight = compute_weight(padded_image, p, q, f, sigma,
h)
                    weights.append(weight)
                    neighborhood_weights.append(padded_image[q[0], q[1]])
            weights = np.array(weights)
            neighborhood_weights = np.array(neighborhood_weights)
            C_p = np.sum(weights)
            filtered_image[i, j] = np.sum(weights * neighborhood_weights)
/ C_p
    return filtered_image.astype(np.uint8)
```

Slika 18 - Funkcija metode nelokalnog usrednjavanja
Izvor: Izradio autor

Funkcija primjenjuje sve korake metode nelokalnog usrednjavanja kako bi uklonila šum sa slike. Funkcija koristi susjedstva za izračun težine i zatim te težine koristi kako bi izračunala novu vrijednost za svaki piksel. Prvi korak funkcije je da pripremi praznu sliku gdje će biti postavljene vrijednosti novih piksela i da na ulaznu sliku doda rub oko slike. Nadalje funkcija iterira kroz sve piksele slike i sprema koordinate trenutnog piksela. Kreiraju se dvije prazne liste za spremanje vrijednosti težina. Zatim funkcija iterira kroz sve piksele u susjedstvu i računa težinu za svaki piksel susjedstva i sprema rezultate u listu. Nakon toga liste se pretvaraju u niz i zbrajaju se sve težine niza.

Zadnji korak funkcije je da se izračuna nova vrijednost piksela koristeći težine i izračunatu sumu, te se na kraju nova slika s novoizračunatim vrijednostima piksela.

Slika 19 prikazuje glavnu funkciju implementacije metode nelokalnog usrednjavanja.

```
def main():
    file_path = choose_image()
    if not file_path:
        print("Nijedna datoteka nije odabrana. Izlazim...")
        return

    image = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
    if image is None:
        print("Učitavanje slike nije uspjelo. Izlazim...")
        return

    noisy_image = add_speckle_noise(image, var=0.2, mean=0.0)

    start_time = time.time()

    sigma = 10.0
    denoised_image = nonlocal_means_filter(noisy_image, f=1, h=0.40*sigma,
sigma=10.0)

    end_time = time.time()
    processing_time = end_time - start_time

    psnr_noisy = calculate_psnr(image, noisy_image)
    ssim_noisy = calculate_ssim(image, noisy_image)

    psnr_denoised = calculate_psnr(image, denoised_image)
    ssim_denoised = calculate_ssim(image, denoised_image)



    print(f'PSNR između originalne slike i slike sa šumom: {psnr_noisy}
dB')
    print(f'SSIM između originalne slike i slike sa šumom: {ssim_noisy}')
```

```
    print(f'PSNR između originalne slike i slike nakon uklanjanja šuma:
{psnr_denoised} dB')
    print(f'SSIM između originalne slike i slike nakon uklanjanja šuma:
{ssim_denoised}')
    print(f'Vrijeme potrebno za uklanjanje šuma metodom nelokalnog
usrednjavanja: {processing_time:.4f} sekundi')

    # Prikazivanje originalne slike, slike sa šumom i slike nakon
uklanjanja šuma
    plt.figure(figsize=(25, 5))

    plt.subplot(1, 4, 1)
    plt.title('Originalna slika')
    plt.imshow(image, cmap='gray')
    plt.axis('off')

    plt.subplot(1, 4, 2)
    plt.title('Slika sa šumom\nPSNR: {:.2f} dB\nSSIM:
{:.4f}'.format(psnr_noisy, ssim_noisy))
    plt.imshow(noisy_image, cmap='gray')
    plt.axis('off')

    plt.subplot(1, 4, 3)
    plt.title('Slika nakon uklanjanja šuma\nPSNR: {:.2f} dB\nSSIM:
{:.4f}'.format(psnr_denoised, ssim_denoised))
    plt.imshow(denoised_image, cmap='gray')
    plt.axis('off')

    plt.subplot(1, 4, 4)
    plt.axis('off')
    plt.text(0.5, 0.5, f'Vrijeme izvođenja:\n{processing_time:.4f}
sekundi',
             horizontalalignment='center', verticalalignment='center',
             fontsize=14, color='black', transform=plt.gca().transAxes)
```

```
    plt.show()


if __name__ == "__main__":
    main()
```

Slika 19 - Glavna funkcija metode nelokalnog usrednjavanja
Izvor: Izradio autor

Funkcija poziva ostale funkcije i odrađuje sve korake od učitavanja slike i dodavanja šuma, pa do uklanjanja šuma metodom. Glavna funkcija je slična kroz sve metode do dijela kada se uklanja šum metodom. Tada se poziva funkcija za tu metodu i odrađuju se potrebni koraci te metode. Prvi koraci su da se učita putanja na računalu do slike i zatim učita slika. Nakon toga na ulaznu sliku se dodaje šum, a nakon dodavanja šuma poziva se funkcija s metodom nelokalnog usrednjavanja kako bi se otklonio taj šum. Nakon izvođenja funkcije s koracima metode računaju se metrike PSNR i SSIM između ulazne slike i slike s dodanim šumom i ulazne slike i slike nakon upotrebe metode. Na kraju se sve slike i svi rezultati metrika prikazuju u zasebnom prozoru.

8. Rezultati

Nakon opisivanja svih odabranih metoda (metoda prostornog usrednjavanja, metoda korištenja valića, metoda neokalnog usrednjavanja) i implementacije metoda u programskom jeziku Python došao je red da se usporede metode i njihovi rezultati. Rezultati metoda bit će uspoređeni na temelju vizualne reprezentacije i na temelju rezultata metrika PSNR i SSIM, a same metode bit će uspoređene uzimajući u obzir nekoliko faktora (kompleksnost metode, kompleksnost implementacije, vrijeme izvođenja koda). Osim što će metode biti ocijenjene na temelju slika gdje je šum naknadno dodan, one će biti primijenjene i na slikama gdje zrnati šum već postoji.

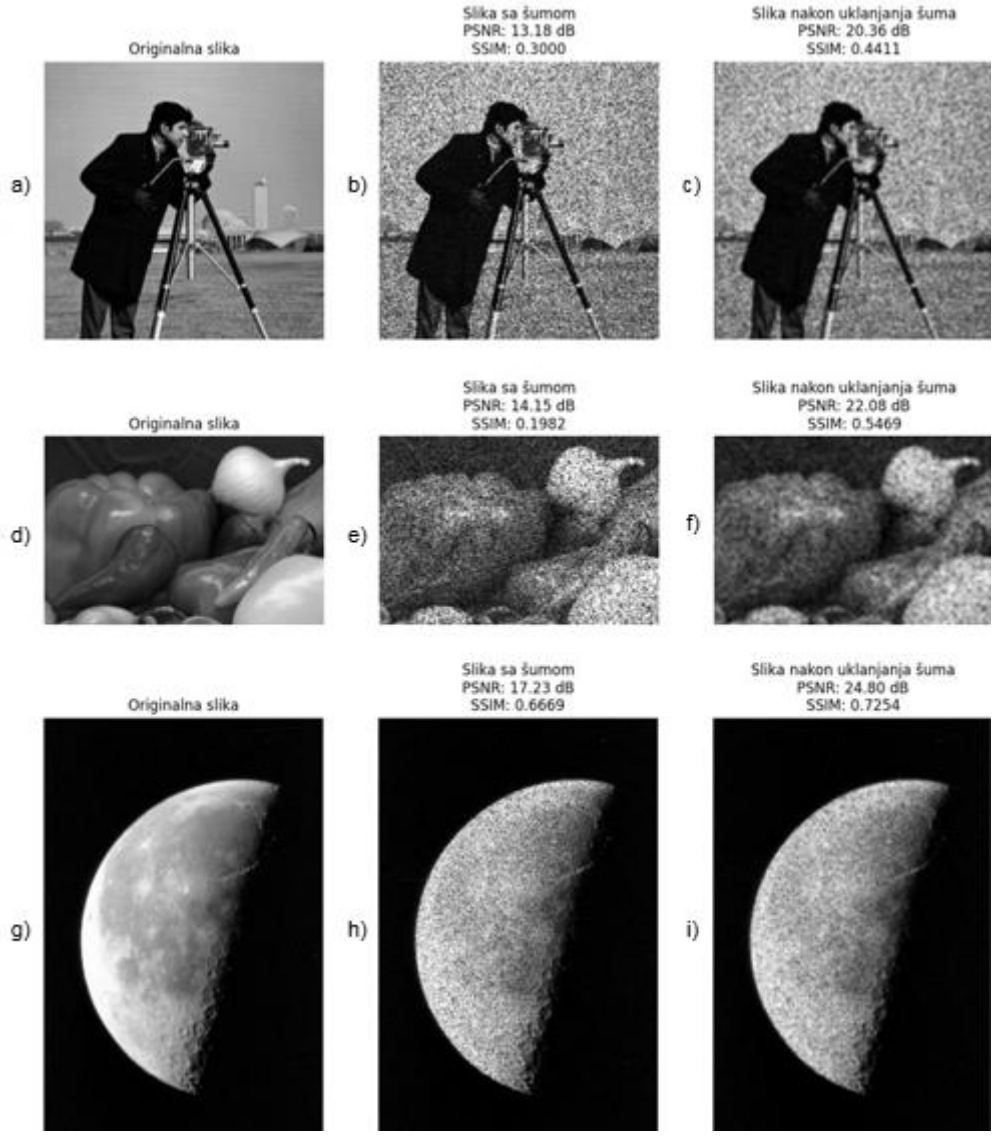
Dodavanje zrnatog šuma napravljeno je pomoću Python biblioteke, a on se dodaje tako da se ulaznoj slici doda Gaussov šum pomnožen s ulaznom slikom. Jačina šuma regulira se varijancom i srednjom vrijednosti Gaussovog šuma. U analizi u sklopu ovog rada varijanca je postavljena na vrijednost 0.2 i srednja vrijednost na 0.0, što predstavlja malo pojačani šum. Testne slike koje su korištene, odnosno slike kojima je dodan šum pa zatim pomoću odgovarajućih metoda uklanjan, prikazane su na Slici 20.

Slika 20 – Testne slike: a) Cameraman, b) Onion, c)Moon
Izvor: Izradio autor

Osim slika prikazanih na Slici 20 koje su korištene tijekom implementacije metoda, metode će se primijeniti i na slikama gdje šum već postoji. To su slike dobivene sonarom, a one su izabrane jer se na tim slikama zrnati šum prirodno pojavljuje.

8.1. Rezultati metode prostornog usrednjavanja

Metoda prostornog usrednjavanja najjednostavnija je za implementaciju od triju navedenih metoda te ima veću brzinu izvođenja, što se može vidjeti u Tablici 3. Sama metoda sastoji se od nekoliko koraka koji ispunjavaju njenu funkcionalnost. Samim time implementacija je vrlo jednostavna, a sam algoritam se vrlo brzo izvodi. Dakle, metoda je „jeftina“, što znači da nije kompleksna za implementirati i ne koristi puno računalnih resursa i vremena. Metoda uzima susjedstvo oko trenutnog piksela, a veličina tog susjedstva u konkretnoj implementaciji postavljena je da bude 3x3 i primijenjena na sve tri testne slike, a rezultati su prikazani na Slici 21.



Slika 21 - Rezultati metode prostornog usrednjavanja na tri testne slike: a) Originalna slika Cameraman, b) Slika Cameraman s dodanim šumom, c) Slika Cameraman nakon uklanjanja šuma, d) Originalna slika Onion, e) Slika Onion s dodanim šumom, f) Slika Onion nakon uklanjanja šuma, g) Originalna slika Moon, h) Slika Moon s dodanim šumom, i) Slika Moon nakon uklanjanja šuma
Izvor: Izradio autor

Slika 21 prikazuje rezultate metode prostornog usrednjavanja. Slike a), d) i g) prikazuju originalnu sliku bez ikakvih promjena, slike b), e) i h) prikazuju kako izgleda originalna slika nakon što smo na nju dodali zrnati šum, a slike c), f) i i) prikazuju kako izgledaju slike nakon što smo zrnati šum pokušali ukloniti korištenjem metode prostornog usrednjavanja. Ako se gledaju slike s više detalja (slika *Cameraman* i slika *Onion*) može se primijetiti da šum nije uklonjen u potpunosti i da ga još dosta ima na slici. Također, promatrajući iste slike primjećuje se značajna zamućenost, pa se samim time teže raspoznaje što je glavni objekt slike i što je u pozadini slike. Ako promatramo jednostavniju sliku (slika *Moon*), odnosno sliku koja nema previše detalja može se primijetiti da je šum bolje uklonjen. Na toj slici detalji nisu toliko potrebni pa se čini da je šum bolje uklonjen i da nema velike zamućenosti.

Tablica 3 prikazuje vrijeme izvođenja metode prostornog usrednjavanja za svaku sliku.

Slike	Vrijeme izvođenja metode u sekundama
Cameraman	0.4656 s
Onion	0.1740 s
Moon	1.3000 s

Tablica 3 - Vrijeme izvođenja metode prostornog usrednjavanja za svaku sliku
Izvor: Izradio autor

Za metrike PSNR i SSIM slike su uspoređene s ulaznom (originalnom) slikom, što je prikazano u Tablici 4, odnosno Tablici 5.

Slike	Slika sa šumom	Slika nakon uklanjanja šuma
Cameraman	13.18 dB	20.36 dB
Onion	14.15 dB	22.08 dB
Moon	17.23 dB	24.80 dB

Tablica 4 - Vrijednosti metrike PSNR za sve slike sa šumom i nakon uklanjanja šuma korištenjem metode prostornog usrednjavanja
Izvor: Izradio autor

Ako pogledamo rezultate metrike PSNR iz Tablice 4, za sliku *Cameraman* gdje je slika c) možemo vidjeti da je rezultat 20.36 dB što je solidan rezultat. Vrijednosti iznad 30 dB ukazuju na sliku visoke kvalitete, dok vrijednosti ispod 20 dB ukazuju na nisku kvalitetu slike i samim time može se zaključiti de rezultat od 20.36 dB nije među boljim rezultatima. Također rezultat PSNR metrike za sliku *Onion* nije najbolji, 20.08 dB. Ako se gleda rezultat za sliku s manje detalja (slika *Moon*) rezultat metrike PSNR je malo veći nego na druge dvije slike, a on iznosi 24.80 dB. Promatrajući rezultate iz Tablice 4 primjećuje se poboljšanje vrijednosti metrike za svaku sliku. Tako je za sliku

Cameraman nakon primjene metode prostornog usrednjavanja vrijednost PSNR-a porasla s 13.18 dB na 20.36 dB, što je poboljšanje od 7.18 dB. Za sliku *Onion* primjenom iste metode vrijednost PSNR-a je porasla s 14.15 dB na 22.08 dB, što je poboljšanje u iznosu od 7.93 dB. Kod zadnje slike, slike *Moon,* također primjenom metode prostornog usrednjavanja primjetan je porast PSNR-a s vrijednosti 17.26 dB na vrijednost 24.80 dB, što je poboljšanje od 7.57 dB.

Slike	Slika sa šumom	Slika nakon uklanjanja šuma
Cameraman	0.3000	0.4411
Onion	0.1982	0.5469
Moon	0.6669	0.7254

Tablica 5 - Vrijednosti metrike SSIM za sve slike sa šumom i nakon uklanjanja šuma korištenjem metode prostornog usrednjavanja
Izvor: Izradio autor

Sljedeća metrika po kojoj se gleda uspješnost metode je SSIM metrika kod koje vrijednosti mogu biti u rasponu od -1 do 1, gdje više vrijednosti pokazuju veći stupanj sličnosti slika. Rezultati metrike SSIM iz Tablice 5 nakon provođenja metode prostornog usrednjavanja za slike *Cameraman* i *Onion* su 0.4411 i 0.5469. Premda se ovi rezultati ne mogu kategorizirati kao najbolji, treba uzeti u obzir da ova metrika uzima više aspekata slike u obzir pa se može utvrditi da su rezultati zadovoljavajući. Ako pogledamo sliku s manje detalja (slika *Moon*) rezultat SSIM metrike je 0.7254 što predstavlja znatno bolji rezultat, odnosno veliki stupanj sličnosti dobivene slike. Ako promatramo Tablicu 5 primjetno je poboljšanje u vrijednostima SSIM metrike. Za sliku *Cameraman* nakon primjene metode prostornog usrednjavanja vrijednost SSIM-a porasla je s 0.3000 na 0.4411, što je poboljšanje od 0.1411. Kod primjene iste metode na sliku *Onion* također je poboljšanje u vrijednostima SSIM.a, pa je tako vrijednost porasla s 0.1982 na 0.5469, što je porast u iznosu od 0.3487. Primjenom metode prostornog usrednjavanja na sliku *Moon* poboljšanje u vrijednosti metrike SSIM je porasla s 0.6669 na vrijednost 0.7254, što je poboljšanje od 0.0564.

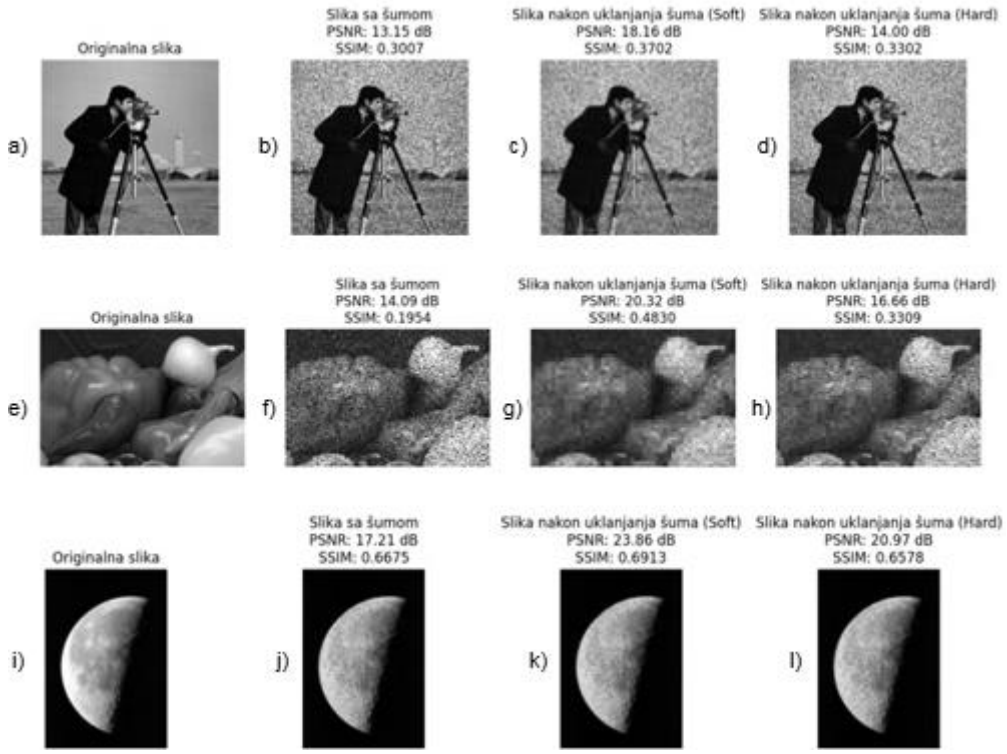
Ako razmotrimo rezultate metrika i vizualne aspekte slike nakon što je na njoj primijenjena metoda prostornog usrednjavanja, može se zaključiti da metoda nije u potpunosti uklonila šum, a sami rezultati se mogu smatrati donekle zadovoljavajućima. Ako gledamo glavne detalje onih slika koje sadrže više detalja (slika *Cameraman* i slika *Onion*), oni su dobro očuvani, ali pozadina slika je znatno zamućena i teško je

razaznati što je u pozadini. U slučaju slike s manje detalja (slika *Moon*), detalji su dobro očuvani i lako se raspoznaje što je na slici.

8.2. Rezultati metode korištenja valića

Metoda korištenja valića najkompleksnija je za implementaciju od triju navedenih metoda. Razlog tome je što metoda radi sa signalima slike. Iako je metoda kompleksna za implementaciju, za izvođenje algoritma nije potrebno previše vremena, ali ipak je potrebno više vremena nego za metodu prostornog usrednjavanja. Metoda koristi više komponenata, a neke bitne koje utječu na rezultat su binarno ograničavanja koje može biti ili meko ili tvrdo i broj razina na koliko metoda radi. Bitno je dobro postaviti ove parametre kako bi rezultat bio što bolji. U implementaciji ovi parametri su postavljeni ovako: binarno ograničavanje koriste se oba načina, a prag je vrijednosti 25 i broj razina na koliko metoda radi je 2. Kao i prijašnja metoda i metoda korištenja valića korištena je na sve tri testne slike i rezultati su prikazani na Slici 22.

Slika 22 prikazuje rezultate metode korištenja valića. Slike a), e) i i) prikazuju originalne slike bez ikakvih promjena, slike b), f), i j) prikazuju kako izgledaju originalne slike nakon što na njih dodamo zrnati šum, slike c), g) i k) prikazuju kako izgledaju slike nakon što smo zrnati šum pokušali ukloniti metodom korištenja valića uz meko binarno ograničavanje, slike d), h) i l) prikazuju kako izgledaju slike nakon što smo zrnati šum pokušali ukloniti istom metodom uz korištenje tvrdog binarnog ograničavanja.



Slika 22 - Rezultati metode korištenja valića na tri testne slike: a) Originalna slika Cameraman, b) Slika Cameraman s dodanim šumom, c) Slika Cameraman nakon uklanjanja šuma uz meko binarno ograničavanje, d) Slika Cameraman nakon uklanjanja šuma uz tvrdo binarno ograničavanje, e) Originalna slika Onion, f) Slika Onion s dodanim šumom, g) Slika Onion nakon uklanjanja šumo uz meko binarno ograničavanje, h) Slika Onion nakon uklanjanja šuma uz tvrdo binarno ograničavanje, i) Originalna slika Moon, j) Slika Moon s dodanim šumom, k) Slika Moon nakon uklanjanja šuma uz meko binarno ograničavanje, l) Slika Moon nakon uklanjanja šuma uz tvrdo binarno ograničavanje
Izvor: Izradio autor

Tablica 6 prikazuje vrijeme izvođenja metode korištenja valića za svaku sliku.

Slike	Vrijeme izvođenja metode u sekundama
Cameraman	0.0126 s
Onion	0.0140 s
Moon	0.0290 s

Tablica 6 - Vrijeme izvođenja metode korištenja valića za svaku sliku
Izvor: Izradio autor

U vrijeme izvođenja metode korištenja valića uključeno je i tvrdo binarno ograničavanje i meko binarno ograničavanje. Razlog toga je što je razlika u vremenu izvođenja pri upotrebi pojedinog binarnog ograničavanja jako malena pa se može i zanemariti.

Ako promatramo slike s više detalja (slika *Cameraman* i slika *Onion*), može se primijetiti da šum nije nabolje uklonjen, ali da nema previše zamućenja na slici. Upotrebom mekog binarnog ograničavanje rezultat je malo bolji nego upotrebom tvrdog binarnog ograničavanja. Kada se pogleda slika s manje detalja (slika *Moon*),

može se vidjeti da je uklanjanje šuma malo bolje, ali opet korištenje mekog binarnog ograničavanja pokazalo se boljom opcijom.

Za metrike PSNR i SSIM kao i kod prijašnje metode, slike su uspoređene s ulaznom (originalnom) slikom, što je prikazano u Tablici 7, odnosno Tablici 8.

Slike	Slika sa šumom	Slika nakon uklanjanja šuma uz meko binarno ograničavanje	Slika nakon uklanjanja šuma uz tvrdo binarno ograničavanje
Cameraman	13.15 dB	18.16 dB	14.00 dB
Onion	14.09 dB	20.32 dB	16.66 dB
Moon	17.21 dB	23.86 dB	20.97 dB

Tablica 7 - Vrijednosti metrike PSNR za sve slike sa šumom i nakon uklanjanja šuma korištenjem metode korištenja valića
Izvor: Izradio autor

Ako promatramo rezultate metriku PSNR iz Tablice 7 za slike s više detalja tada su rezultati za sliku *Cameraman* 18.16 dB i 14.00 dB, a za sliku *Onion* 20.32 dB i 16.66 dB. Kod ovih rezultata također se može primijetiti da tamo gdje je primijenjeno meko binarno ograničavanje rezultat je bolji, a to su rezultati 18.16 dB za sliku *Cameraman* i 20.32 dB za sliku *Onion*. Rezultati su jako nisko, pa čak ispod 20 dB što znači da su slike loše kvalitete s obzirom na ulaznu sliku. Kada se promatra slika s manje detalja (slika *Moon*), rezultat metrike PSNR je 23.86 dB i 20.97 dB. Opet je bolji rezultat tamo gdje je primijenjeno meko binarno ograničavanje, 23.86 dB. Od svih rezultata metrike jedino za rezultat 23.86 dB može se reći da je zadovoljavajući. Primjenom metode korištenja valića poboljšale su se vrijednosti metrike PSNR. Tako su se vrijednosti PSNR metrike za sliku *Cameraman* porasle s vrijednosti 13.15 dB na 18.16 dB primjenom mekog binarnog ograničavanja i 14.00 dB primjenom tvrdog binarnog ograaničavanja. Primjenom metode korištenja valića uz meko binarno ograničavanje vrijednost metrike PSNR porasla je za 5.01 dB, a primjenom metode uz tvrdo binarno ograničavanje vrijednost je porasla za 0.85 dB. Primjenom metode korištenja valića na slici *Onion* vrijednosti metrike su porasle s vrijednosti 14.09 dB na vrijednost 20.32 dB uz primjenu mekog binarnog ograničavanja i 16.66 dB uz primjenu tvrdog binarnog ograničavanja. Vrijednost PSNR metrike primjenom metode korištenja valića uz meko binarno ograničavanje porasla je za 6.23 dB, dok je primjenom iste metode uz tvrdo binarno ograničavanje vrijednost PSNR metrike porasla za 2.57 dB. Također primjenom metode korištenja valića na sliku *Moon* vrijednosti metrike PSNR porasle su s vrijednosti 17.21 dB na vrijednost 23.86 dB uz korištenje mekog binarnog

ograničavanja i 20.97 dB uz korištenje tvrdog binarnog ograničavanja. Primjenom metode uz korištenje mekog binarnog ograničavanja vrijednost PSNR metrike porasla je za 6.65 dB i primjenom metode uz korištenje tvrdog binarnog ograničavanja vrijednost PSNR metrike je porasla za 3.76 dB.

Slike	Slika sa šumom	Slika nakon uklanjanja šuma uz meko binarno ograničavanje	Slika nakon uklanjanja šuma uz tvrdo binarno ograničavanje
Cameraman	0.3007	0.3702	0.3302
Onion	0.1954	0.4830	0.3309
Moon	0.6675	0.6913	0.6578

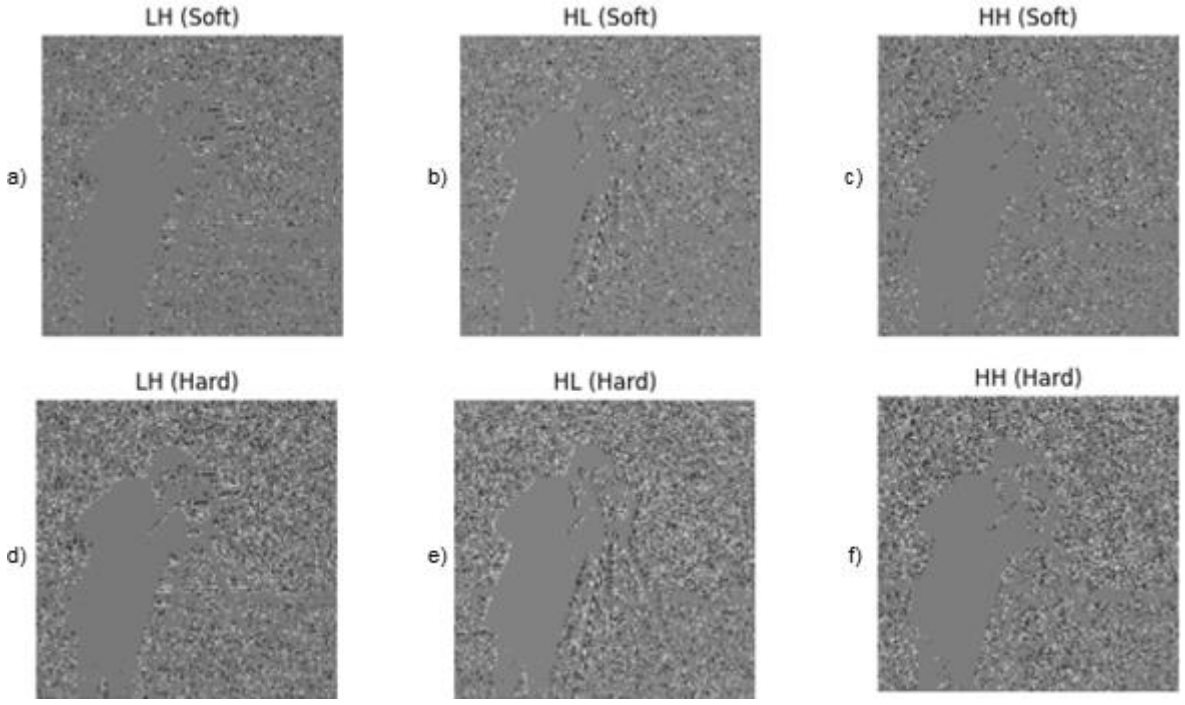
Tablica 8 - Vrijednosti metrike SSIM za sve slike sa šumom i nakon uklanjanja šuma korištenjem metode korištenja valića
Izvor: Izradio autor

Kad se promatraju rezultati SSIM metrika iz Tablice 8, rezultati također nisu najbolji. Za slike s više detalja, kao što su slika *Cameraman* i slika *Onion* rezultati metrike su 0.3702, 0.3302 i 0.4860, 0.3309. Ovdje je također primjetno da tamo gdje je korišteno meko binarno ograničavanje rezultati su bolji, 0.3702 za sliku *Cameraman* i 0.4860 za sliku *Onion*. Iako rezultati izgledaju dobro s obzirom na raspon metrike, oni su i dalje loši, jer što su rezultati bliži 0 to znači da slike nisu slične. Ako se promatra slika s manje detalja (slika *Moon*), rezultati metrike su 0.6913, 0.6578 i također je rezultat bolji kada je primijenjeno meko binarno ograničavanje, 0.6913. Primjenom metode korištenja valića na slici *Cameraman* vrijednosti SSIM metrike su porasle s vrijednosti 0.3007 na vrijednost 0.3702 uz primjenu mekog binarnog ograničavanja i uz primjenu tvrdog binarnog ograničavanja vrijednost je porasla na vrijednost 0.3302. Upotrebom metode korištenja valića uz primjenu mekog binarnog ograničavanja vrijednost SSIM metrike porasla je za 0.0695, a upotrebom metode korištenja valića uz primjenu tvrdog binarnog ograničavanja vrijednost SSIM metrike porasla je za 0.0295. Rezultati SSIM metrike za sliku *Onion* uz primjenu metode korištenja valića porasli su s vrijednosti 0.1954 na vrijednost 0.4860 uz primjenu mekog binarnog ograničavanja i uz primjenu tvrdog binarnog ograničavanja vrijednost SSIM metrike porasla je na 0.3309. Primjenom metode korištenja valića uz upotrebu mekog binarnog ograničavanja na sliku *Onion* vrijednost metrike SSIM porasla je za 0.2876 i primjenom metode korištena valića uz upotrebu tvrdog binarnog ograničavanja vrijednost je porasla za 0.1355. Kada smo metodu korištenja valića primijenili na slici *Moon* vrijednosti metrike SSIM porasla je s vrijednosti 0.6675 na vrijednost 0.6913 uz primjenu mekog binarnog

ograničavanja i uz primjenu tvrdog binarnog ograničavanja vrijednost metrike SSIM smanjila se na vrijednost 0.6578. Primjenom metode korištenja valića uz upotrebu mekog binarnog ograničavanja na sliku *Moon* vrijednost je porasla za 0.0238, a primjenom metode korištenja valića uz upotrebu tvrdog binarnog ograničavanja vrijednost SSIM metrike smanjila se za 0.0097.

Ako promatramo rezultate metrika i vizualne aspekte slike nakon što je korištena metoda korištenja valića, može se zaključiti da nije u potpunosti uklonila zrnati šum, a rezultati su lošiji nego nakon korištenja prijašnje metode. Kada bi gledali bitne detalje slike na slikama s više detalja (slika *Cameraman* i slika *Onion*), oni su donekle očuvani, ali ono što je u pozadini slike se jedva vidi. Slika s manje detalja (slika *Moon*) dobro je očuvana i što je na njoj se lako može raspoznati.

Prednost ove metode je što može iz slike izvući njene horizontalne detalje (LH), vertikalne detalje (HL) i dijagonalne detalje (HH) i to čini jako dobro, a kako oni izgledaju prikazano je na Slici 23.



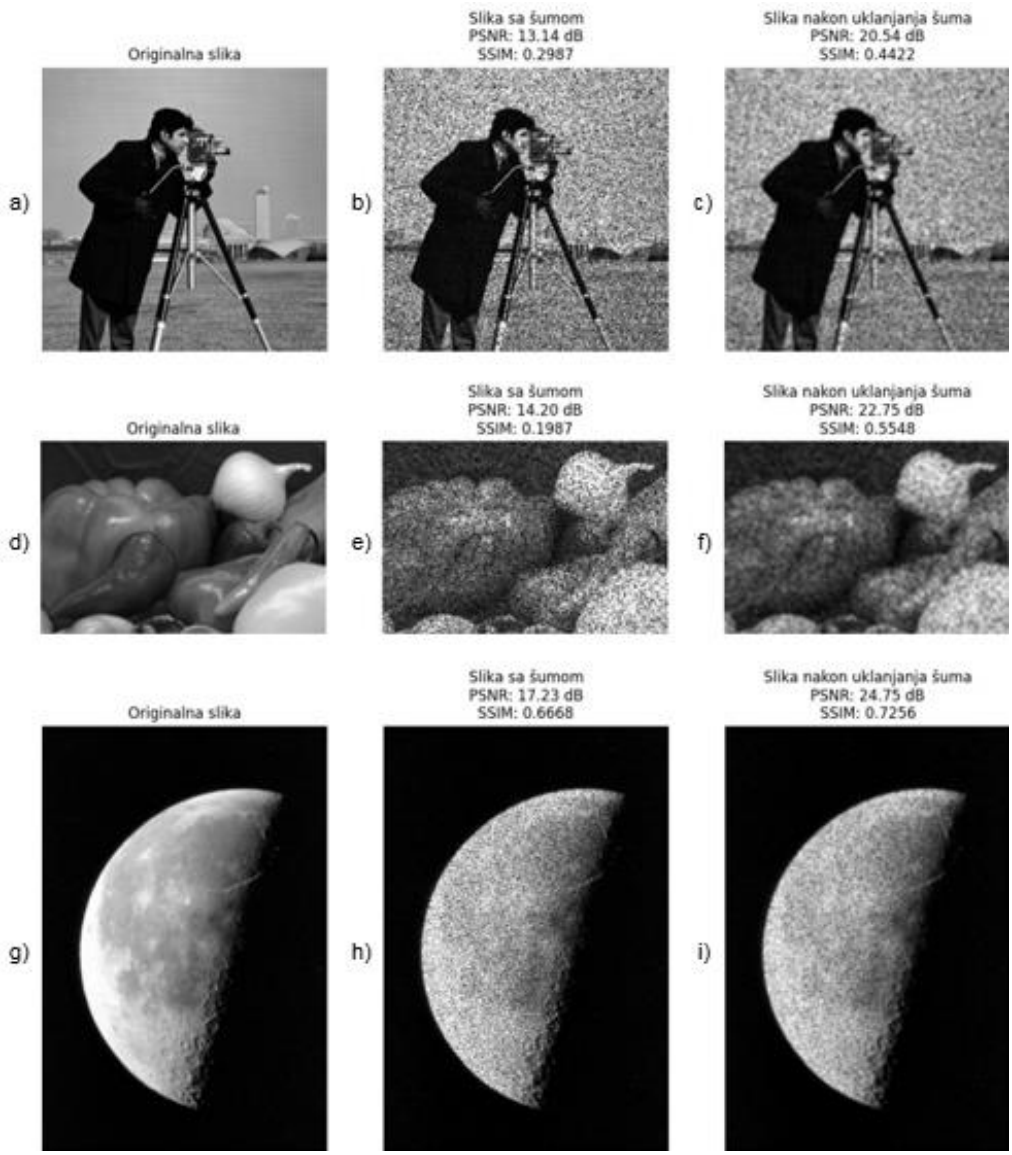
Slika 23 - Izdvojeni detalji slike Cameraman pomoću metode korištenja valića: a) Horizontalni detalji uz meko binarno ograničavanje, b) Vertikalni detalji uz meko binarno ograničavanje, c) Dijagonalni detalji uz meko binarno ograničavanje, d) Horizontalni detalji uz tvrdo binarno ograničavanje, e) Vertikalni detalji uz tvrdo binarno ograničavanje, f) Dijagonalni detalji uz tvrdo binarno ograničavanje
Izvor: Izradio autor

Slika 23 prikazuje izdvojene komponente pri korištenju određenog binarnog ograničavanja. U ovom slučaju primjetno je da kada se koristi tvrdo binarno ograničavanje vidljivost detalja je veća i lakše ih je primijetiti. Ovi detalji su izvučeni iz slike *Cameraman* što se može primijetiti jer negdje se može vidjeti i tronožac kamere. Za dobivanje jasnih pojedinih komponenti bolje je koristiti manju razinu na kojoj metoda radi, pa je tako za primjer na Slici 23 korištena razina vrijednosti 1.

8.3. Rezultati metode nelokalnog usrednjavanja

Metoda nelokalnog usrednjavanja je od triju navedenih metoda srednje kompleksnosti za implementaciju. Algoritmu za izvođenje potrebno je dosta vremena, što ju čini metodom koja se najdulje izvodi. Samim time za ovu metodu potrebno je više resursa i više vremena, što se može vidjeti u Tablici 9. Metoda ima nekoliko parametara koji se trebaju postavljati za dobivanje optimalnih rezultata, a oni su: broj piksela koji se uzima oko odabranog piksela, standardna devijacija šuma i parametar filtriranja šuma. U implementaciji broj piksela koji se uzima oko odabranog piksela je 1, standardna devijacija šuma je postavljena na 10.0 i parametar filtra šuma je 0.40*10.0 što je prikazano u Tablici 1 za vrijednost standardne devijacije šuma. Rezultati dobiveni primjenom metode nelokalnog usrednjavanja na testnim slikama prikazani su na Slici 24.

Slika 24 prikazuje rezultate nakon što je na slikama korištena metoda nelokalnog usrednjavanja. Slike a), d) i g) prikazuju originalnu sliku bez ikakvih promjena, slike b), e) i h) prikazuju kako izgledaju originalne slike nakon što je na njih dodan zrnati šum, slike c), f) i) prikazuju kako izgledaju slike nakon što smo šum pokušali ukloniti korištenjem metode nelokalnog usrednjavanja.



Slika 24 - Rezultati metode nelokalnog usrednjavanja na tri testne slike: a) Originalna slika Cameraman, b) Slika Cameraman s dodanim šumom, c) Slika Cameraman nakon uklanjanja šuma, d) Originalna slika Onion, e) Slika Onion s dodanim šumom, f) Slika Onion nakon uklanjanja šuma, g) Originalna slika Moon, h) Slika Moon s dodanim šumom, i) Slika Moon nakon uklanjanja šuma
Izvor: Izradio autor

Tablica 9 prikazuje vrijeme izvođenja metode nelokalnog usrednjavanja za sve slike.

Slike	Vrijeme izvođenja metode u sekundama
Cameraman	56.6575 s
Onion	22.3657 s
Moon	196.1686 s

Tablica 9 - Vrijeme izvođenja metode nelokalnog usrednjavanja za svaku sliku
Izvor: Izradio autor

Na slikama s više detalja (slika *Cameraman* i slika *Onion*) može se primijetiti da šum nije u potpunosti uklonjen, ali da nema prevelikog zamućenja kao kod prijašnjih metoda. Na slici *Cameraman* nakon korištenja metode pozadinski detalji su također

malo više vidljivi. Ako promatramo sliku s manje detalja (slika *Moon*) zrnati šum je dobro uklonjen i nema zamućenja slike.

Za metrike PSNR i SSIM kao i kod prijašnjih metoda, slike su uspoređene s ulaznom (originalnom) slikom, što je prikazano u Tablici 10, odnosno Tablici 11.

Slike	Slika sa šumom	Slika nakon uklanjanja šuma
Cameraman	13.14 dB	20.54 dB
Onion	14.20 dB	22.75 dB
Moon	17.23 dB	24.75 dB

Tablica 10 - Vrijednosti metrike PSNR za sve slike sa šumom i nakon uklanjanja šuma korištenjem metode nelokalnog usrednjavanja
Izvor: Izradio autor

Ako promatramo rezultate PSNR metrike iz Tablice 10 za slike s više detalja (slika *Cameraman* i slika *Onion*) oni su bolji nego prijašnje dvije metode. Rezultati PSNR metrike su za sliku *Cameraman* 20.54 dB i za sliku *Onion* 22.75 dB. Ovi rezultati mogu se svrstati u pozitivne rezultate jer označava dobru kvalitetu slike. Ako promatramo rezultat za sliku s manje detalja (slika *moon*), 24.75 dB. Primjenom metode nelokalnog usrednjavanja na sliku *Cameraman* vrijednost metrike PSNR porasla je s vrijednosti 13.14 dB na vrijednost 20.54 dB, što je povećanje u vrijednosti od 7.4 dB. Ako metodu nelokalnog usrednjavanja primijenimo na slici *Onion* vrijednost metrike također poraste s vrijednosti 14.20 dB na vrijednost 22.75 dB, što je povećanje od 8.55 dB. Na slici *Moon* primjenom metode nelokalnog usrednjavanja također dobijemo povećanje u vrijednosti PSNR metrike, vrijednost se poveća s vrijednosti 17.23 dB na vrijednost 24.75 dB, što je povećanje od 7.52 dB.

Slike	Slika sa šumom	Slika nakon uklanjanja šuma
Cameraman	0.2987	0.4422
Onion	0.1987	0.5548
Moon	0.6668	0.7256

Tablica 11 - Vrijednosti metrike SSIM za sve slike sa šumom i nakon uklanjanja šuma korištenjem metode nelokalnog usrednjavanja
Izvor: Izradio autor

Kada promatramo rezultate metriku SSIM iz Tablice 11, rezultati su također zadovoljavajući. Za slike s više detalja rezultati su: za sliku *Cameraman* 0.4422 i za sliku *Onion* 0.5548. Ako promatramo sliku s manje detalja rezultat metrike SSIM je znatno veći, 0.7256. Ovako velik rezultat znači da su slike jako slične. Primjenom metode nelokalnog usrednjavanja na sliku Cameraman vrijednosti SSIM metrike je

porasla s vrijednosti 0.2987 na vrijednost 0.4422, što je povećanje u vrijednosti od 0.1435. Ako metodu nelokalnog usrednjavanja primijenimo na sliku *Onion* vrijednosti SSIM metrike također porastu i to s vrijednosti 0.1987 na vrijednost 0.5548, što je povećanje od 0.3561. Kada na sliku *Moon* primijenimo metode nelokalnog usrednjavanja vrijednost SSIM metrike porast s vrijednosti 0.6668 na vrijednost 0.7256, što je povećanje vrijednosti od 0.0588.

Ako promatramo rezultate metrika i vizualne aspekte slike nakon što je korištena metoda nelokalnog usrednjavanja, može se zaključiti da korištenjem metode nije uklonjen sav šum, ali slika nije ni previše zamućena. Kada se gledaju bitni detalji slike na slikama s više detalja oni su ostali dobro očuvani, pa čak i neki pozadinski detalji su bolje očuvani. Na slici s manje detalja, svi detalji su ostali očuvani nakon korištenja metode.

8.4. Usporedba metoda

Metode će biti uspoređene na temelju rezultata metrika i vizualnih aspekata, a za usporedbu ćemo koristiti sve tri slike *Cameraman, Onion, Moon*. Usporedba performansi pojedinih metoda primijenjenih na tri testne slike dana je u obliku vrijednosti poboljšanja metrika PSNR i SSIM u Tablicama 12, 13 i 14.

Metode	Poboljšanje PSNR-a	Poboljšanje SSIM-a
Metoda prostornog usrednjavanja	7.18 dB	0.1411
Metoda korištenja valića (meko binarno ograničavanje)	5.01 dB	0.0695
Metoda korištenja valića (tvrdo binarno ograničavanje)	0.85 dB	0.0295
Metoda nelokalnog usrednjavanja	**7.40 dB**	**0.1435**

Tablica 12 - Vrijednosti poboljšanja metrika PSNR i SSIM slike Cameraman nakon primjene metoda
Izvor: Izradio autor

Metode	Poboljšanje PSNR-a	Poboljšanje SSIM-a
Metoda prostornog usrednjavanja	7.93 dB	0.3487
Metoda korištenja valića (meko binarno ograničavanje)	6.23 dB	0.2876
Metoda korištenja valića (tvrdo binarno ograničavanje)	2.57 dB	0.1355
Metoda nelokalnog usrednjavanja	**8.55 dB**	**0.3561**

Tablica 13 - Vrijednosti poboljšanja metrika PSNR i SSIM slike Onion nakon primjene metoda
Izvor: Izradio autor

Metode	Poboljšanje PSNR-a	Poboljšanje SSIM-a
Metoda prostornog usrednjavanja	**7.57 dB**	0.0564
Metoda korištenja valića (meko binarno ograničavanje)	6.65 dB	0.0238
Metoda korištenja valića (tvrdo binarno ograničavanje)	3.76 dB	0.0097
Metoda nelokalnog usrednjavanja	7.52 dB	**0.0588**

Tablica 14 - Vrijednosti poboljšanja metrika PSNR i SSIM slike Moon nakon primjene metoda
Izvor: Izradio autor

Promatrajući vizualne aspekte slika može se reći da slika nakon korištenja metode prostornog usrednjavanja ima najveće zamućenje, dok slika nakon korištenja metode korištenja valića ima najmanje zamućenje. Nadalje, glavni aspekti slike kod svake metode ostaju poprilično isti, jedino se pozadinski detalji slike najbolje vide pri korištenju metode nelokalnog usrednjavanja. Ako se gleda samo vizualno, teško je reći koja metoda je nabolje rekonstruirala sliku i poslije koje metode je slika najsličnija originalnoj slici. Za to ocijeniti korištene su metrike PSNR i SSIM. PSNR metrika ocjenjuje kvalitetu rekonstruirane slike i rezultat je u dB (decibelima), a SSIM metrika ocjenjuje sličnosti između dvije slike i uzima u obzir svjetlinu, kontrast i strukturalnu sličnost.

Promatranjem Tablice 12, Tablice 13 i Tablice 14, primjetno je da je metoda nelokalnog usrednjavanja dala najbolje rezultate praktički na sve tri testne slike, i u pogledu metrike PSNR i metrike SSIM. Kao što se može vidjeti iz Tablice 12, za sliku *Cameraman* metoda nelokalnog usrednjavanja dala je najbolje rezultate u pogledu

poboljšanja vrijednosti metrike PSNR, gdje to poboljšanje iznosi 7.40 dB. S druge strane, metoda korištenja valića, neovisno o vrsti korištenog binarnog ograničavanja, dala je najlošije rezultate. Ako nastavimo promatrati rezultate poboljšanja metrike PSNR iz tablica za sliku *Onion* (Tablica 13), može se primijetiti da je metoda nelokalnog usrednjavanja ponovno dala najbolji rezultat, koji iznosi 8.55 dB. Kao i na prethodnoj slici, metoda korištenja valića dala je najlošije rezultate. Promatranjem rezultata poboljšanja metrike PSNR za sliku *Moon* (Tablica 14) primjetna je mala razlika u odnosu na prijašnje slike. Kod slike *Moon* najbolji rezultat poboljšanja PSNR metrike imala je metoda prostornog usrednjavanja, a on je iznosio 7.57 dB. Ovaj rezultat neznatno je bolji od rezultata metode nelokalnog usrednjavanja koja je na prijašnjim slikama pokazala se bolja, a koji je iznosio 7.52 dB. Kao i na prijašnjim slikama metoda korištenja valića imala je najlošije rezultate poboljšanja metrike PSNR.

U istim tablicama su i rezultati poboljšanja metrike SSIM, koji nisu puno drugačiji od rezultata poboljšanja metrike PSNR. Ako promatramo rezultate iz Tablice 12, Tablice 13 i Tablice 14 za metriku SSIM, možemo primijetiti da metoda nelokalnog usrednjavanja ima najbolje rezultate poboljšanja na sve tri slike, dok metoda korištenja valića na sve tri slike daje najlošije rezultate. Za sliku *Cameraman (Tablica 12)* rezultat poboljšanja SSIM metrike za metodu nelokalnog usrednjavanja iznosio je 0.1435. Kod rezultata za sliku *Onion* (Tablica 13) metoda nelokalnog usrednjavanja također dala najbolji rezultat, a on iznosi 0.3561. Za sliku *Moon* (Tablica 14) metoda nelokalnog usrednjavanja imala je najbolji rezultat SSIM metrike, koji je iznosio 0.0588.

Iz ovih rezultata primjetno je da je metoda nelokalnog usrednjavanja imala najbolje rezultate poboljšanja metrika PSNR i SSIM, dok je metoda korištenja valića imala najlošije rezultate na svakoj slici. Iako je na slici *Moon* metoda prostornog usrednjavanja imala najbolji rezultat poboljšanja vrijednosti metrike PSNR, rezultat metode nelokalnog usrednjavanja bio je jako blizu. Metoda korištenja valića imala je najlošije rezultate poboljšanja vrijednosti metrika, ali ova metoda je na sliku nakon uklanjanja šuma dodala najmanje zamućenja temeljem provedene vizualne analize.

Osim što metode možemo usporediti prema rezultatima poboljšanja vrijednosti metrika PSNR i SSIM, možemo ih usporediti i po vremenu izvođenja. Tablica 15 prikazuje vremena izvođenja svake od metoda na pojedinoj testnoj slici.

Metode	*Cameraman*	*Onion*	*Moon*
Metoda prostornog usrednjavanja	0.4656 s	0.1740 s	1.3000 s
Metoda korištenja valića	**0.0126 s**	**0.0140 s**	**0.0290 s**
Metoda nelokalnog usrednjavanja	56.6575 s	22.3657 s	196.1686 s

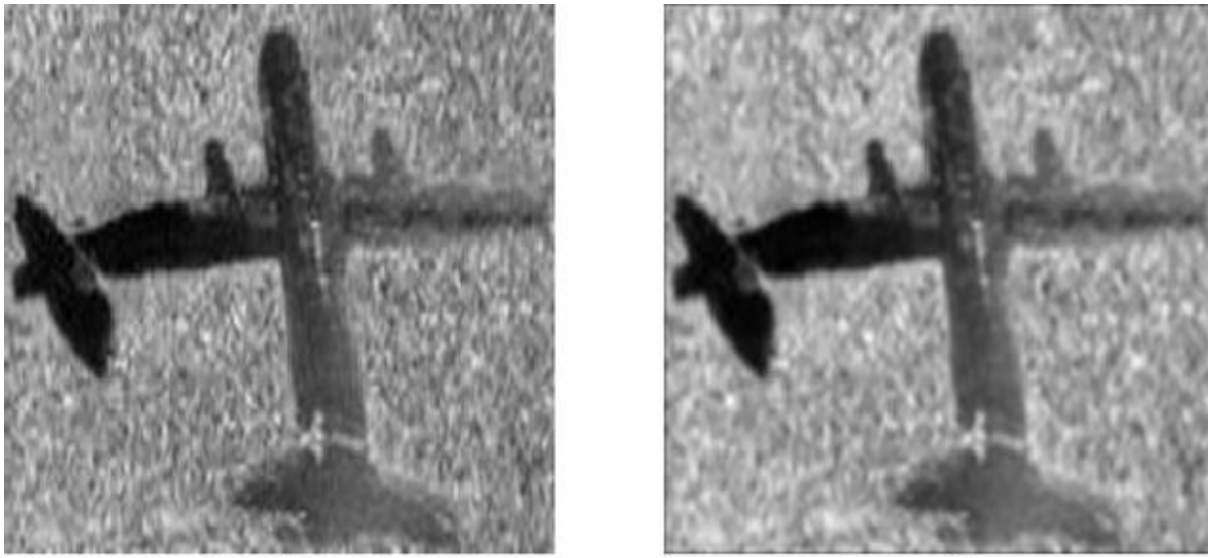
Tablica 15 - Vrijeme izvođenja metoda za svaku testnu sliku
Izvor: Izradio autor

Promatranjem Tablice 15 možemo primijetiti da metoda korištenja valića, koja je imala najlošije rezultate poboljšanja vrijednosti metrika PSNR i SSIM, ima najbrže vrijeme izvođenja za svaku testnu sliku. Vrijeme izvođenja metode korištenja valića za sliku *Cameraman* iznosilo je 0.0126 sekundi, za sliku *Onion* iznosilo je 0.0140 sekundi, a za sliku *Moon* vrijeme izvođenja bilo je 0.0290 sekundi. Osim toga primjetno je da metoda nelokalnog usrednjavanja koja je imala najbolje rezultate poboljšanja vrijednosti metrika PSNR i SSIM, ima znatno veće vrijeme izvođenja od ostalih metoda.

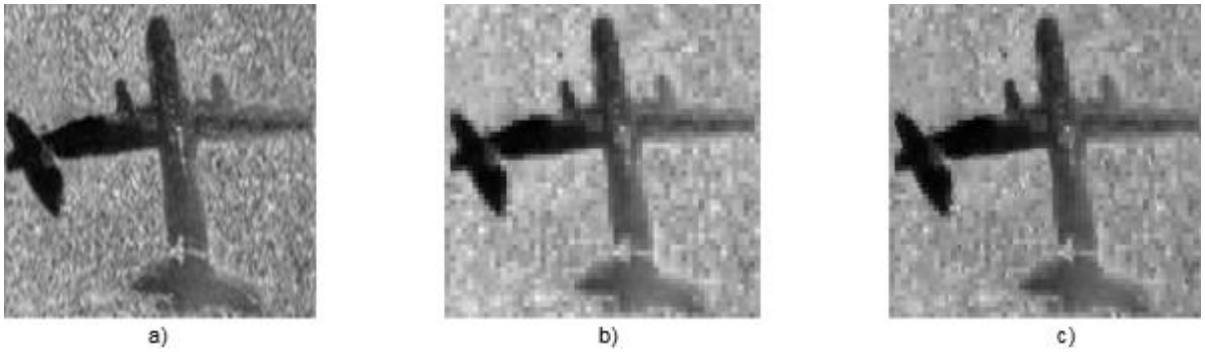
8.5. Upotreba metoda na slikama gdje je zrnati šum prirodna pojava

Osim na testnim slikama metode ćemo isprobati na slikama gdje je zrnati šum prirodna pojava, slike sonara dobivene eksperimentalnim mjerenjima u stvarnim uvjetima. Za ovu potrebu koristiti ćemo slike iz javno dostupnog skupa podataka s takvim slikama (huoguanying, 2020).

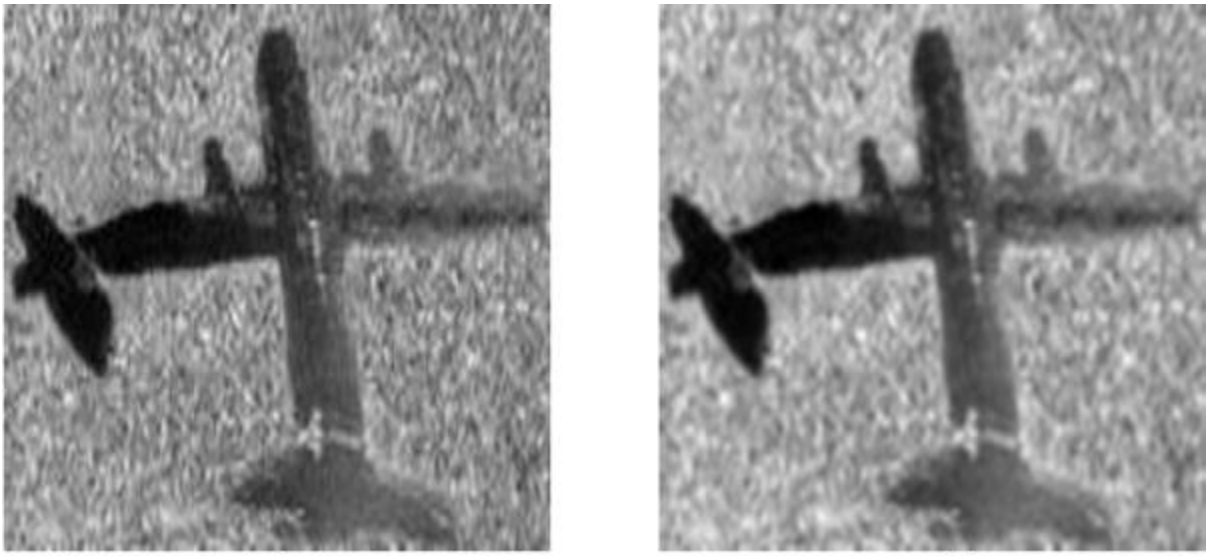
Slika 25, Slika 26 i Slika 27 prikazuju upotrebu pojedine metode na istoj slici na kojoj zrnati šum prirodno postoji.



Slika 25 - Upotreba metode prostornog usrednjavanja na slici 19 iz javno dostupnog skupa podataka sa slikama s postojećim zrnatim šumom: a) Originalna slika 19 iz skupa podataka, b) Slika 19 nakon uklanjanja šuma
Izvor: Izradio autor



Slika 26 - Upotreba metode korištenja valića na slici 19 iz javno dostupnog skupa podataka sa slikama s postojećim zrnatim šumom: a) originalna slika 19 iz skupa podataka, b) Slika 19 nakon uklanjanja šuma uz meko binarno ograničavanje, c) Slika 19 nakon uklanjanja šuma uz tvrdo binarno ograničavanje
Izvor: Izradio autor



Slika 27 - Upotreba metode nelokalnog usrednjavanja na slici 19 iz javno dostupnog skupa podataka sa slikama s postojećim zrnatim šumom: a) Originalna slika 19 iz skupa podataka, b) Slika 19 nakon uklanjanja šuma
Izvor: Izradio autor

Kada se slika vizualno razmatra, glavni detalj na slici, koji je avion, je ostao nepromijenjen, dok se oko njega tekstura malo promijenila. Zrnati šum sliku malo zatamni što se može primijetiti na originalnim slikama, a na slikama kada su primijenjene metode primjetno je da su teksture oko aviona svijetlije i malo manje pikselizirane. Na ovim slikama, kao i na testnim slikama, primjetno je da metode nisu u potpunosti uklonile šum, ali da su sačuvale kvalitetu bitnih dijelova slike.

9. Zaključak

U ovom radu proučavane su tri različite metode za uklanjanje multiplikativnog šuma iz digitalnih slika, a korištene metode su: metoda prostornog usrednjavanja, metoda korištenja valića i metoda nelokalnog usrednjavanja.

Metoda prostornog usrednjavanja najjednostavnija je i najučinkovitija u smislu vremena i obrade. U ovoj metodi se uzima prosječna vrijednost piksela u lokalnom susjedstvu. Rezultati pokazuju da, iako može smanjiti šum, ova metoda ima tendenciju zamućenja slike, posebno u finim područjima. Rezultati poboljšanja vrijednosti metrike PSNR bile su jako blizu najboljim rezultatima, a na jednoj slici je najbolji rezultat. Na slici *Cameraman* vrijednost poboljšanja iznosila je 7.18 dB, na slici *Onion* poboljšanje je iznosilo 7.93 dB, a na slici *Moon* gdje je ta vrijednost bila najbolja iznosila je 7.57 dB. Rezultati poboljšanja vrijednosti metrike SSIM za sve slike bili su drugi najbolji rezultati. Za sliku *Cameraman* ta vrijednost poboljšanja iznosila je 0.1411, za sliku *Onion* vrijednost poboljšanja metrike iznosila je 0.3487 i za sliku *Moon* vrijednost poboljšanja iznosila je 0.0564.

Metoda korištenja valića najkompleksnija je od sve tri metode, kako sa stajališta implementacije tako i računanja. Radi na način da transformira sliku u valićnu domenu, primjenjuje prag na valićnim koeficijentima i zatim sliku transformira natrag kako je bila. Ova metoda je dobra u očuvanju detalja slike, ali nije najbolja za potpuno uklanjanje šuma iz slike. Prema rezultatima poboljšanja metrika PSNR i SSIM, ova metoda pokazala se najlošijom. Rezultati poboljšanja PSNR metrike za sliku *Cameraman* iznosili su: 5.01 dB i 0.85 dB. Za liku *Onion* metoda je opet imala najlošije rezultate, a iznosili su: 6.23 dB i 2.57 dB. Za sliku *Moon* vrijednosti poboljšanja PSNR metrike iznosile su: 6.65 dB i 3.76 dB. Rezultati poboljšanja vrijednosti za metriku SSIM također su najlošiji rezultati za sve slike. Za sliku *Cameraman* vrijednosti poboljšanja SSIM metrike iznosile su: 0.0695 i 0.0295. Rezultati poboljšanja vrijednosti metrike SSIM za sliku Onion iznosili su: 0.2876 i 0.1355. Za sliku *Moon* vrijednosti poboljšanja SSIM metrike iznosile su: 0.0238 i 0.0097. Iako je ova metoda imala najlošije vrijednosti poboljšanja metrika nakon uklanjanja šuma, nakon uklanjanja šuma ona je temeljem provedene vizualne analize dodala najmanje zamućenja na sliku u odnosu na ostale metode.

Metoda nelokalnog usrednjavanja koristi koncept da se pikseli nelokalnog susjedstva mogu koristiti za uklanjanje šuma. Više je vremenski i resursno zahtjevna, ali daje dobar balans između smanjenja šuma i očuvanja detalja. Ova metoda imala je najbolje rezultate poboljšanja vrijednosti metrika PSNR i SSIM. Samo na jednoj slici, slici *Moon*, metoda prostornog usrednjavanja imala je bolji rezultat, ali nije bio značajno veći. Rezultat poboljšanja vrijednosti PSNR metrike za sliku *Cameraman* je 7.4 dB. Za sliku *Onion* vrijednost poboljšanja vrijednosti metrike PSNR iznosila je 8.55 dB. Rezultat poboljšanja vrijednosti metrike PSNR za sliku *Moon* iznosio je 7.52 dB. Ako promatramo rezultate poboljšanja vrijednosti metrike SSIM, ova metoda na svim slikama ima najbolje rezultate. Za sliku *Cameraman* rezultat poboljšanja vrijednosti metrike SSIM iznosio je 0.1435. Rezultat poboljšanja vrijednosti SSIM metrike za sliku *Onion* iznosio je 0.3561. Za sliku *Moon* poboljšanje vrijednosti SSIM metrike iznosilo je 0.0588.

Gledajući sveukupno metode, metoda nelokalnog usrednjavanja bila je najbolja za poboljšanje vrijednosti metrika PSNR i SSIM, osim na slici *Moon* gdje je metoda prostornog usrednjavanja imala malo bolji rezultat poboljšanja metrike PSNR. Metoda nelokalnog usrednjavanja bila je najbolja zato što balansira smanjenje šuma i očuvanje detalja. Ako se gleda kompleksnost, metoda korištenja valića, iako je kompleksna i računski zahtjevna nije se pokazala tako dobrom u smislu smanjenja šuma u usporedbi s očekivanjima. To ovu metodu čini nepoželjnom. Gledajući s pozicije implementacije, metoda prostornog usrednjavanja je najjednostavnija tehnika, a korisna je kada je potrebno brzo i jednostavno uklanjanje šuma po cijenu zamućenja. Za brzo i jednostavno uklanjanje šuma preporučuje se metoda prostornog usrednjavanja, jer je laka za implementaciju i brza. U situacijama gdje su detalji slike primarne važnosti preporuča se korištenje metode nelokalnog usrednjavanja.

10. Literatura

Adobe, 2024. *What is noise in photography?.* [Mrežno]
Available at: https://www.adobe.com/creativecloud/photography/hub/guides/what-is-noise-in-photography.html
[Pokušaj pristupa 20 lipanj 2024].

Analytics India magazine PVT & AIM Media house LLC, 2024. *Different Types of Noises and Image Denoising Methods.* [Mrežno]
Available at: https://analyticsindiamag.com/ai-mysteries/a-guide-to-different-types-of-noises-and-image-denoising-methods/#h-speckle-noise
[Pokušaj pristupa 20 lipanj 2024].

ASU News, 2023. *Advances in polarized imaging reveal new visual capabilities.* [Mrežno]
Available at: https://news.asu.edu/20230928-advances-polarized-imaging-reveal-new-visual-capabilities#:~:text=One%20way%20to%20see%20beyond,of%20an%20image%20to%20create
[Pokušaj pristupa 21 lipanj 2024].

Bnou, K., Raghay, S. & Hakim, A., 2020. *SpringerOpen.* [Mrežno]
Available at: https://asp-eurasipjournals.springeropen.com/articles/10.1186/s13634-020-00693-4
[Pokušaj pristupa 24 lipanj 2024].

Boyat, A. K. & Joshi, B. K., 2015. A review papper: Noise models in digital image processing. *Signal & Image Processing : An International Journal (SIPIJ) Vol.6, No.2*, travanj, pp. 63-75.

Buades, A., Coll, B. & Morel, J. M., 2011. *IPOL Journal · Image Processing On Line.* [Mrežno]
Available at: https://www.ipol.im/pub/art/2011/bcm_nlm/?utm_source=doi
[Pokušaj pristupa 2 srpanj 2024].

Dan, S. & Zhimin, Z., 2019. *Multiplicative Model and Filtering of Speckle Image in Lidar System.* [Mrežno]
Available at: https://webofproceedings.org/proceedings_series/ECS/ISMHI%202019/SMH103.pdf
[Pokušaj pristupa 21 lipanj 2024].

GeeksforGeeks, 2024. *OpenCV Tutorial in Python.* [Mrežno]
Available at: https://www.geeksforgeeks.org/opencv-python-tutorial/
[Pokušaj pristupa 24 lipanj 2024].

Gonzalez, S., 2023. *Salvator Labs.* [Mrežno]
Available at: https://blog.salvatorelabs.com/understanding-the-median-filter-a-powerful-tool-for-noise-reduction-in-image-processing/
[Pokušaj pristupa 21 lipanj 2024].

huoguanying, 2020. *GitHub.* [Mrežno] Available at: https://github.com/huoguanying/SeabedObjects-Ship-and-Airplane-dataset
[Pokušaj pristupa 13 ožujak 2024].

LOPAC, N., 2022. *Predavanja iz kolegija Digitalna obrada i analiza slike.* s.l.:Sveučilište Jurja Dobrile u Puli, Tehnički fakultet.

Matplotlib development team, 2024. *matplotlib.pyplot.* [Mrežno] Available at: https://matplotlib.org/stable/api/pyplot_summary.html
[Pokušaj pristupa 24 lipanj 2024].

National instruments corp , 2024. *Peak Signal-to-Noise Ratio as an Image Quality Metric.* [Mrežno] Available at: https://www.ni.com/en/shop/data-acquisition-and-control/add-ons-for-data-acquisition-and-control/what-is-vision-development-module/peak-signal-to-noise-ratio-as-an-image-quality-metric.html
[Pokušaj pristupa 22 lipanj 2024].

PyData Sphinx Theme, 2024. *NumPy documentation.* [Mrežno] Available at: https://numpy.org/devdocs/
[Pokušaj pristupa 24 lipanj 2024].

Python Software Foundation, 2024. *Graphical User Interfaces with Tk.* [Mrežno] Available at: https://docs.python.org/3/library/tk.html
[Pokušaj pristupa 24 lipanj 2024].

scikit-image team, 2024. *scikit-image.* [Mrežno] Available at: https://scikit-image.org/docs/stable/api/api.html
[Pokušaj pristupa 22 lipanj 2024].

Singh, P. & Pandey, R. S., 2016. *ResearchGate.* [Mrežno] Available at: https://www.researchgate.net/publication/312061215_Speckle_noise_Modelling_and_implementation
[Pokušaj pristupa 21 lipanj 2024].

Wikipedia, 2024. *Structural similarity index measure.* [Mrežno] Available at: https://en.wikipedia.org/wiki/Structural_similarity_index_measure
[Pokušaj pristupa 22 lipanj 2024].

11. Popis slika

12. Popis tablica

13. Prilozi

Poveznica na github repozitorij s pohranjenim programskim kodom za implementaciju metoda analiziranih u diplomskom radu:

- LukaErcegovac/Diplomski (github.com)

Implementacija metoda je odrađena tri puta: samostalnom implementacijom koraka svake metode, korištenjem biblioteka i samostalnom implementacijom koraka, ali bez dodavanja šuma na sliku kako bi se metoda mogla primjeniti na slikama gdje šum već postoji. Tijekom implementacije korišteno je virtualno okruženje (*venv*) i sve korištene biblioteke spremljene su u dokument "*requirements.txt*".

Sve je strukturirano na način da svaki dokument s imenom metode u sebi sadrži sva tri načina implementacije, a to izgleda ovako:

mean-filter:

- *mean-filter-library.py* → implementacija metode lokalnog usrednjavanja korištenjem biblioteke, korištena zbog testiranja samostalne implementacije.
- *mean-filter-without-adding-noise.py* → samostalna implementacija koraka metode lokalnog usrednjavanja bez dodavanja šuma, koristi se za slike gdje šum već postoji.
- *mean-filter.py* → samostalna implementacija koraka metoda lokalnog usrednjavanja.

non-local-mean-filter:

- non-local-mean-filter-library.py → implementacija metode nelokalnog usrednjavanja korištenjem biblioteke, korištena zbog testiranja samostalne implementacije.
- *non-local-mean-filter-without-adding-noise.py* → samostalna implementacija koraka metode nelokalnog usrednjavanja bez dodavanja šuma, koristi se za slike gdje šum već postoji.
- *non-local-mean-filter.py* → samostalna implementacija koraka metode nelokalnog usrednjavanja.

wavelet-filter:

- *wavelet-filter-library.py* → implementacija metode korištenja valića korištenjem biblioteke, korištena zbog testiranja samostalne implementacije.
- *wavelet-filter-without-adding-noise.py* → samostalna implementacija koraka metode korištenja valića bez dodavanja šuma, koristi se za slike gdje šum već postoji.
- *wavelet-filter.py* → samostalna implementacija koraka metode korištenja valića.

Korištenje i pokretanje koda

Ako se želi koristiti virtualno okruženje, potrebno ga je instalirati. Virtualno okruženje može se koristiti kako instalirane biblioteke ne bi bile instalirane lokalno i jednom kada se dokument izbriše i one će biti izbrisane.

Instaliranje virtualnog okruženja:

- pip install virtualenv
- python -m venv <ime virtualnog okruženja>

Virtualno okruženje može se pokrenuti ovako:

- cd .\venv\Scripts\
- .\activate

Ako se pojavljuje greška, onda je prije pokretanja druge naredbe (".\activate") potrebno pokrenuti sljedeću naredbu:

- Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass

Kada se pokrene virutalno okruženje, potrebno je vratiti se natrag u glavni dokument, a to se može postići naredbom:

- cd .. → naredbu je potrebno pokrenuti dva puta

Bitno je da se instaliraju sve potrebne biblioteke za rad, a one se nalaze u "requirements.txt":

- pip install requirements.txt

Nakon što su instalirane sve potrebne biblioteke može se odabrati koju metodu se želi koristiti, a kao što je prije objašnjeno implementacija svake metode je u zasebnom dokumentu. Prvo je potrebno da se odabere dokument metode koja se želi koristiti, a to se može napraviti pomoću naredbe "cd":

- cd <ime dokumenta>

Nakon što je odabran dokument Python, dokumenti u njemu se mogu pokrenuti pomoću sljedeće naredbe:

- python <ime Python dokumenta>.py

Sažetak

U ovom radu obrađene su metode za uklanjanje multiplikativnog šuma iz digitalnih slika. Multiplikativni šum je kompleksan zrnati šum koji se obično nalazi na slikama dobivenim putem specifičnih tehnologija snimanja kao što su ultrazvuk, radar i lasersko skeniranje. Ovaj tip šuma značajno otežava vizualnu interpretaciju i obradu slike. Cilj rada je istražiti i usporediti učinkovitost triju odabranih metoda: metode prostornog usrednjavanja, metode korištenja valića i metode nelokalnog usrednjavanja u smanjenju multiplikativnog šuma uz očuvanje bitnih detalja. Pritom je prikazan opis matematičkih modela i teorijskih temelja pojedine metode. Sve metode implementirane su u programskom jeziku Python, a njihova učinkovitost evaluirana je na temelju vizualnih aspekata i pomoću dviju metrika: vršnog omjera signala i šuma (PSNR) i indeksa strukturalne sličnosti (SSIM).

Dobiveni rezultati pokazali su da svaka metoda ima svoje prednosti i nedostatke. Metoda prostornog usrednjavanja jednostavna je za implementaciju, ali nedostatak joj je što znatno zamućuje sliku. Metoda korištenja valića omogućila je bolju očuvanost detalja temeljem vizualne analize, ali je na svim slikama dala najlošije rezultate poboljšanja vrijednosti metrika. Ova metoda također je i najkompleksnija za implementaciju. Metoda nelokalnog usrednjavanja računalno je najzahtjevnija, ali je kontinuirano postizala najbolje rezultate u očuvanju strukture slike.

Na temelju analize provedene u radu može se zaključiti da izbor optimalne metode uklanjanja šuma ovisi o karakteristikama slike, šumu i osobnim potrebama. Kroz rad je prikazana detaljna analiza i usporedba metoda, što doprinosi razumijevanju problema i tehnika uklanjanja multiplikativnog šuma iz digitalnih slika.

Ključne riječi:

multiplikativni šum, digitalna obrada slike, prostorno usrednjavanje, valićna transformacija, nelokalno usrednjavanje, zrnati šum

Abstract

This paper addresses methods for removing multiplicative noise from digital images. Multiplicative noise is a complex grainy noise typically found in images obtained through specific imaging technologies such as ultrasound, radar, and laser scanning. This type of noise significantly complicates visual interpretation and processing of the image. The aim of the thesis is to investigate and compare the effectiveness of three selected methods: the spatial averaging method, the wavelet method, and the non-local means method in reducing multiplicative noise while preserving important details. A description of the mathematical models and theoretical foundations of each method is also presented. All methods were implemented in the Python programming language, and their effectiveness was evaluated based on visual aspects and using two metrics: peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM).

The results showed that each method has its advantages and disadvantages. The spatial averaging method is simple to implement, but its drawback is that it significantly blurs the image. The wavelet method allowed for better preservation of details based on visual analysis, but provided the worst results in terms of metric improvements across all images. This method is also the most complex to implement. The non-local means method is the most computationally demanding, but consistently achieved the best results in preserving image structure.

Based on the analysis conducted in this thesis, it can be concluded that the choice of the optimal noise removal method depends on the characteristics of the image, the noise, and individual needs. The thesis provides a detailed analysis and comparison of the methods, contributing to the understanding of the problem and techniques for removing multiplicative noise from digital images.

Keywords:

multiplicative noise, digital image processing, spatial averaging, wavelet transformation, non-local averaging, speckle noise