Izveštaj drugog domaćeg zadatka

iz predmeta

Digitalna obrada slike

Katarina Petrović 2021/0068

1. Zadatak:

U slici koju je neophodno obraditi uočen je problem do kog dolazi u procesu **polutoniranja**. Prikazom slike i njenog spektra uočeno je da postoje komponente u spektru koje se periodično ponavljaju i koje su posledica toga što je slika puna sitnijih i krupnijih crnih tačkica postavljenih periodično (u formatu kvadratne rešetke).

Postupak obrade ove slike zasniva se prvenstveno na **niskofrekventnom filtriranju** (s obzirom da je spektar „originalne slike“ najvećim delom prisutan samo na niskim učestanostima), a zatim na filtriranju **Notch filtrom** – filtrom koji pogađa frekvencije u spektru koje su izmeštene iz centra slike i potiskuje ih (kao niskofrekventno filtriranje samo sa izmeštenim centrom). Nakon frekvencijskog filtriranja ovim filtrima, slika ostaje samo sa neophondim komponentama u spektru koje sadrže informaciju o sadržaju slike.

Za realizaciju ove dve funkcije slike, realizovane su python funkcije notch i lpfilter. Ukratko, ove funkcionalnosti mogu biti opisane sledećim algoritmima (kod i komentari delova koda su u Jupyter Notebook-u):

Notch filtar:

Opisi ulaznih parametara: filt_type – tip filtra (idealni, Gaus-ov ili Batervort-ov), notch – pass ili reject (da li zelimo propusnik ili nepropusnik delova spektra), Ny i Nx – dimenzije slike (spektra), C – pozicije centara na kojima zelimo da vrsimo filtriranje, sigma_x i sigma_y – standardna devijacija po x i y osi u slucaju Gaus-ovog filtra (moze se prevesti i na druge tipove filtara ali nema potrebe), n – red filtra (potrebno za Batervort-ov filtar)

Postupak:

- Pomerimo koordinate slike tako da je vrednost (0,0) u centru slike

- Za svaki element iz vektora C vrsimo narednu obradu:
    - Formiramo komplementarni centar trenutnom centru C (da bi filtar bio centralno simetrican)
    - Formiramo D0 i D0c (na osnovu x0, y0, x0c i y0c) – rastojanje svih tacaka u spektru od pocetnog i komplementarnog centra
    - Biramo na osnovu tipa filtra filter masku – na osnovu D0/D0c ili X0, Y0/X0c, Y0c i funkcije filter maske svi pikseli (u spektru) dobijaju određenu vrendost
- Na osnovu parametra notch biramo da li želimo pass ili reject funkcionalnost
- Filter masku vraćamo kao izlaz funkcije

Niskofrekventni filtar (lpfilter):

Opisi ulaznih parametara: filt_type – tip filtra (idealni, Gaus-ov ili Batervort-ov), Ny i Nx – dimenzije slike (spektra), sigma_x i sigma_y – standardna devijacija po x-osi i po y-osi (za Gaus-ov filtar), n – red filtra (za Batervort-ov filtar)

Postupak:

- Pomerimo koordinate slike tako da je vrednost (0,0) u centru slike
- Biramo na osnovu tipa filtra filter masku – na osnovu D0/D0c ili X0, Y0/X0c, Y0c i funkcije filter maske svi pikseli (u spektru) dobijaju određenu vrendost
- Filter masku vraćamo kao izlaz funkcije

Ostatak koda vezanog za ovaj zadatak predstavlja primenu ovog filtra na primeru slike girl_ht.gif. Konkretno, slika je prvenstveno učitana kao float u opsegu 0-1, potom je **generisan njen spektar** (pomoću np.fft.fft2 i np.fft.fftshift). Zatim je primenjen **lpfilter** (Gaus-ov tip filtra kako ne bi imali odraze sinc funkcije kasnije u prostornom domenu kao kod primene idealnog filtra, standardna devijacija je 120 po obe ose), a potom **notch filtar** sa centrima na svakih Nx/16 i Ny/16 (određeno na osnovu spektra slike). U okviru notch filtra, šum na visokim frekvencijama potisnut je Gausovim filtrom standardnih devijacija 30 i 25. Nakon ovih operacija, slika je vraćena u prostorni domen.

Prethodno su prikazane ulazna slika i njen spektar, a naredne slike predstavljaju Notch filtar koji je generisan u svim tačkama u kojima se javlja šum usled polutoniranja, kao i spektar izlazne slike (na kojoj su primenjeni prvo lpfilter, a potom notch filter).

Poređenje ulazne i izlazne slike prikazano je u nastavku:

Izlaznoj slici je moguće **popraviti kontrast** kako bi se detalji na slici videli još bolje, međutim to nije specificirano postavkom zadatka, pa nije u ovom slučaju odrađeno.

2. Zadatak:

Postupak restauracije date slike je sledeći:

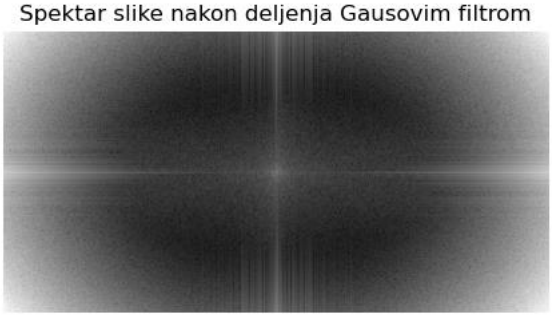
- Razdvojiti sliku na **R, G i B kompoentu** i naredne korake primenjivati na svakoj komponenti
- Prikazati spektar slike i u njemu uočiti tip šuma – u ovom slučaju šum je **Gaus-ov** (rečeno postavkom zadatka), standardne devijacije (procenjeno) **60** po horizontalnoj osi i **45** po vertikalnoj osi

- Generisati isti takav šum – za ovo je korišćena funkcija **lpfilter** iz prethodnog zadatka koja generiše 2D Gaus-ov šum na osnovu zadatih dimenzija slike (spektra) i standardnih devijacija po x i y osi (funkciji su prosleđene standardne devijacije 60 i 45)

- Spektar **slike podeliti generisanim šumom** – na ovaj način nestaju komponente u spektru koje je prethodno stvarao šum, ali se pojavljuju visoke vrednosti na delovima gde je generisani šum imao jako male vrednosti (ovo su uglavnom visoke učestanosti)
- Kako bi se rešio problem vrednosti bliskih 1 koje su nastale kao posledica ovog deljenja, primenjujemo **Viner-ov filtar**, tako da je rezultujući spektar:

$$img_{fft,new} = \frac{img_{fft}}{h} \cdot \frac{h^2}{(h^2 + k)}$$

  Gde je desni množilac W aktivan u slučaju visokih vrednosti, što definišemo zavisno od parametra k – što je k veće to ćemo više kažnjavati visoke vrednosti (prag je nizak), za manje k ovaj prag je veći pa ćemo manji broj visokih vrednosti izmeniti ovim filtrom
- U ovom slučaju, optimalno **k je 0.0002**



- Ako posmatramo sve tri komponente slike, konkretno u ovom slučaju je najbolje odšumljena komponenta slike **R**, pa se na osnovu nje mogu najbolje pročitati tablice automobila (tablice su **AWCA-510**), ali je svakako neophodno kombinovati sve tri komponente slike kako bi se dobila restaurirana polazna slika

Izlazna slika - R komponenta

Izlazna slika - G komponenta

Izlazna slika - B komponenta

- Kombinovati sve tri komponente slike koristeći np.stack funkciju

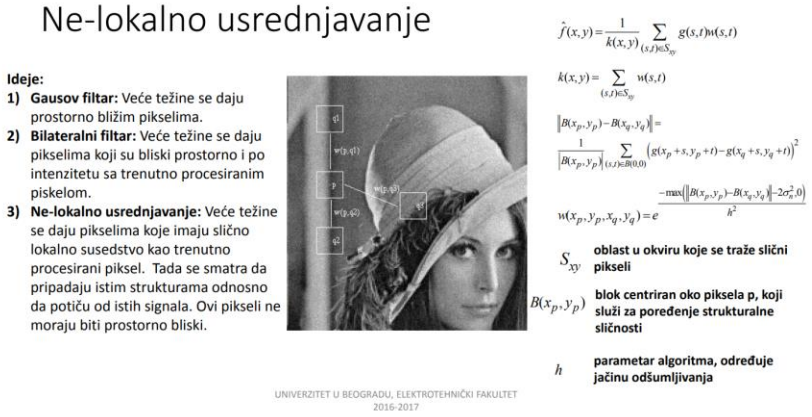
Uporedni prikaz početne i restaurirane slike prikazan je u nastavku:

Odavde je takođe moguće očitati tablice **AWCA-510** (ali značajno teže nego posmatranjem odvojenih RGB komponenti). Generalno, sliku je moguće dodatno izoštriti ili poboljšati kontrast kako bi se bolje očitale tablice, međutim od tih testiranih metoda, nijedna nije značajno poboljšala restauraciju slike u kontekstu očitavanja tablica (a da pri tom nije narušila kvalitet ostatka slike).

3. Zadatak:

I. Realizacija dos_non_local_means

Funkcija dos_non_local_means realizovana je prema definiciji sa slajda:



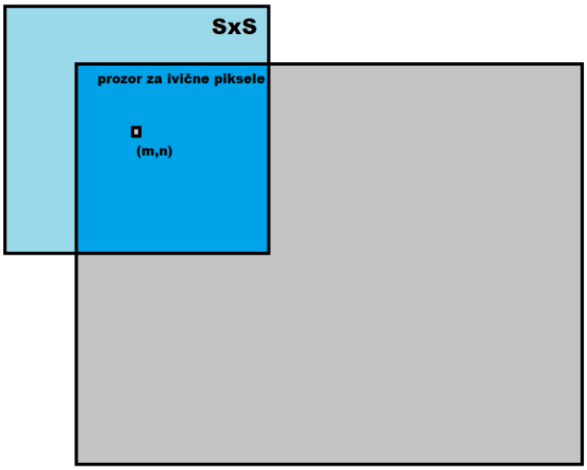
Prvenstveno su za ulaznu sliku definisani **radius_mask** i **radius_win** – radius_mask definiše parametar K (veličina prozora koji predstavlja lokalno susedstvo svakog piksela), a radius_win parametar S (veličina prozora u okviru kog se traže pikseli sa sličnim lokalnim susedstvom kao trenutno posmatrani piksel).

Zatim se računa **prva norma** $|B(x_p, y_p)|$ koja predstavlja broj elemenata u masci KxK.

Slika se proširuje za radius_mask vrednost sa svake strane kako bi svaki piksel slike imao svoje lokalno susedstvo (program bi bio složeniji ukoliko bi proširivali za vrednost radius_win jer bi dimenzije prozora bile nepotrebno velike u slučaju ivičnih piksela – obrađivali bi se pikseli koji u suštini ne postoje na slici i uticali bi na računanje težina, a nisu relevantni). Kasnije će biti objašnjeno računanje prozora i kako je ono implementirano u slučaju piksela blizu ivica.

Zatim se za svaki piksel računa njegovo **lokalno susedstvo** pomoću funkcije **view_as_windows** – rezultat je promenljiva windows koja je tipa matrica dimenzija istih kao dimenzije slike ali je svaki element matrice lokalno susedstvo piksela koji se nalazi na toj poziciji na slici.

Glavna petlja funkcije prolazi kroz svaki piksel na slici (m – za iteraciju po vrstama, n – za iteraciju po kolonama) i prvenstveno definiše prozor u okviru kog će tražiti piksele za upoređivanje sa pikselom na poziciji (m,n). Za ne-ivične piksele, prozor je dimenzija SxS i predstavlja okolinu oko piksela radijusa radius_win. Za piksele blizu ivice, ne postoje svi pikseli tako da se uzme prozor dimenzija SxS, pa je prozor za pretragu piksela smanjen na sledeći način:



Prozor za pretragu je **tamno plave boje**.

Prozor za pretragu definišu promenljive pocetak_prozora_vrsta, …, kraj_prozora_kolona, a sam prozor je promenljiva prozor_kratak. Takođe, definisane su iste promenljive, samo na proširenoj slici, kako bi mogli da posmatramo lokalna susedstva svih piksela na slici. Na ovaj način je dobijena promenljiva **prozor**.

U okviru prozora formiramo od piksela **njihova lokalna susedstva** funkcijom sliding_window_view i čuvamo ih u promenljivoj search_windows.

Računamo **drugu normu** i težine za svaki piksel po definiciji.

U izlaznoj slici, piksel na poziciji (m,n) dobija vrednost u skladu sa **intenzitetima piksela** u okviru prozora za pretragu i **težinama** koje odgovaraju sličnosti tih piksela sa trenutno posmatranim.

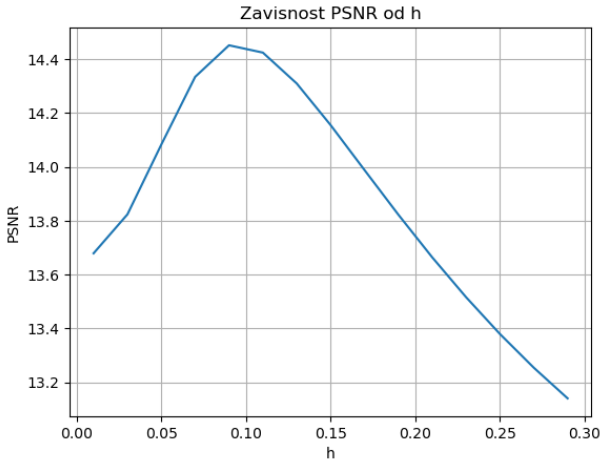
Ovime je završena funkcija dos_non_local_means.

II. Računanje PSNR

Takođe, realizovana je i funkcija calculate_psnr koja na osnovu **nezašumljene** slike i **zašumljene/obrađene** slike računa PSNR.

III. Testiranje funkcija i pronalaženje optimalnih parametara

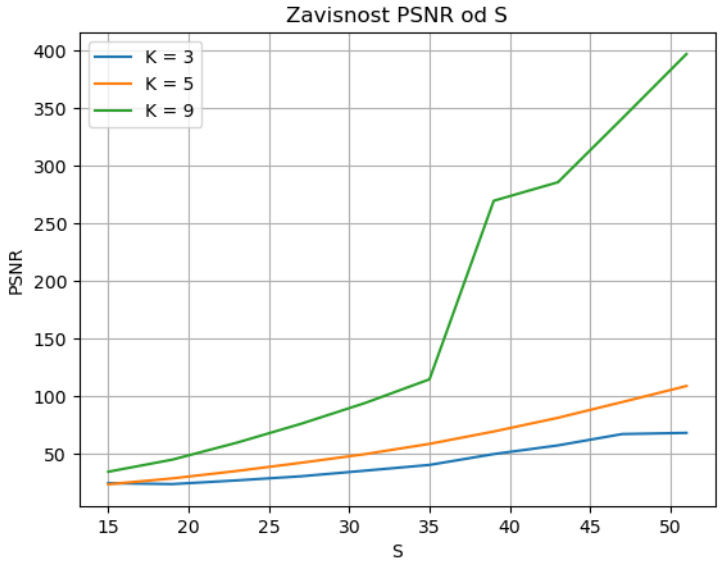
Kako bi odredili pogodnu vrednost parametra h, funkcija dos_non_local_means je testirana sa parametrima K = 5 i S = 15 i posmatrana je promena vrednosti PSNR sa promenom parametra h. Uočeno je da se za h=0.09 dobija maksimalno PSNR (za date vrednosti K i S).



Naredne slike prikazuju kako izgleda slika pre i nakon primene funkcije dos_non_local_means sa ovim parametrima.

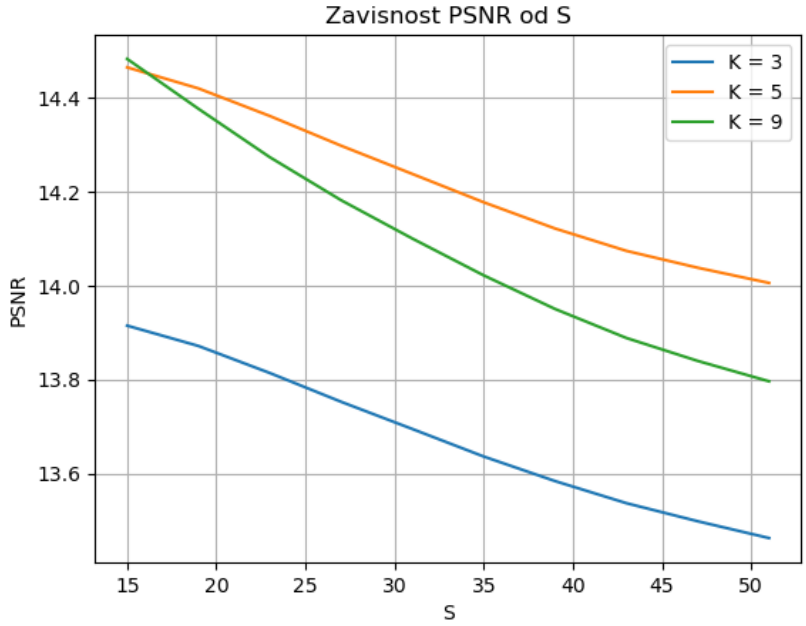
Takođe, prikazano je i kako izgleda izlazna slika u slučaju različitih vrednosti K i S (po zahtevu zadatka). Primećujemo da je najbolji PSNR u slučaju K = 9, S = 15.

Zavisnost PSNR i vremena izvršavanja funkcije dos_non_local_means prikazani su na sledećim graficima. Ovi grafici su realizovani na sledeći način: Posmatrane su 3 vrednosti parametra K (3, 5 i 9), dok je vrednost S uzimana iz opsega 15-51 sa korakom 4. Za svaku kombinaciju ovih parametara, računata je **PSNR vrednost** i **vreme izvršavanja** funkcije, koji su sačuvani u matricama 3x10 (matrica_vreme i matrica_psnr) gde je svaka vrsta vezana za jednu vrednost K.



Sa grafika je moguće primetiti da vreme izvršavanja funkcije raste sa porastom S i sa porastom K što je očekivano jer porastom veličine prozora (i po S i po K) imamo veći broj piksela koje treba uvrstiti u račun. Teorijski, ova zavisnost je kvadratna (i po S i po K), odnosno ukupna složenost funkcije je $O(M \cdot N \cdot K^2 \cdot S^2)$, gde su M i N dimenzije slike.

Sa stanovišta brzine izvršavanja funkcije, uočljivo je da odabir vrednosti K = 9 značajno odstupa od vrednosti K = 5 i K = 3, dok razlika ove dve vrednosti nije značajna. Pri izboru vrednosti S, najbolje je birati što manju vrednost kako bi se smanjila brzina izvršavanja funkcije.



Sa grafika zavisnosti PSNR od vrednosti S, uočljivo je da PSNR opada sa porastom vrednosti S (slični pikseli u okviru prozora imaju manje težine jer u okviru tog prozora postoji i dosta onih

piksela koji nisu slični trenutno posmatranom pikselu). Takođe, uočljivo je da je PSNR dosta mali za K = 3, pa u odabiru optimalnih parametara ne treba uzeti tu vrednost.

Na osnovu ovih zavisnosti, zaključak je da su optimalni parametri za obradu slike funkcijom dos_non_local_means K = 5 i S = 15. Za ove vrednosti, brzina izvršavanja je relativno mala (20ak sekudni), a odnos signal/šum je veliki (14.45). Ovaj rezultat je već prethodno prikazan iznad (prilikom ispitivanja optimalnog h).