Fakultet elektrotehnike, računarstva i informacijskih tehnologija
Osijek

Obrada slike i računalni vid

SEMINARSKI RAD

Algoritmi zaglađivanje slika (engl. image smoothing)

Anto Tufeković

Osijek, 2020.

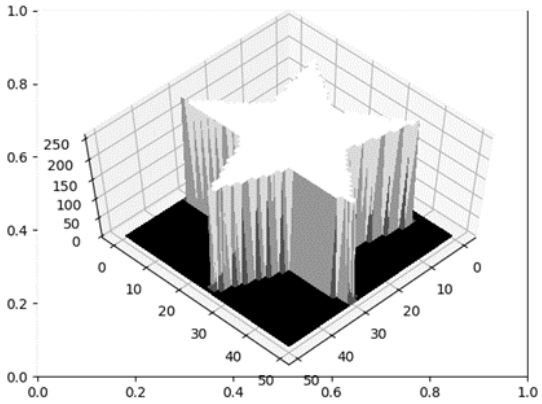
Sadržaj

1. Kratki uvod

Zaglađivanje slika (engl. *Image smoothing*) je postupak mijenjanja slike tako da svi pikseli međusobno imaju manje oštar prijelaz boja. To se koristi za slučajeve kad se nad slikama moraju obavljati ostale transformacije koje se baziraju na derivativnim postupcima, jer izglađivanjem se oštri prijelazi što stabilizira derivacije nad tim mjestima, jer derivacije nad mjestima sa velikim promjenama(npr. šum, oštri rubovi/prijelazi) poprime visoke vrijednosti ili jednostavno da bi maknuli šum iz zadanih slika za bolju preglednost.

Primjerice, uzmimo sliku zvijezde (slika 1.1.). Kad prikažemo sliku u trodimenzionalnom obliku koristeći python funkcije dobijemo sliku 1.2., na kojoj se vidi da postoji oštar prijelaz između vrijednosti bijele i crne boje.

Slika 1.1. - 50x50 slika zvijezde



Slika 1.2. - prikaz zvijezde sa plot_surface()

Cilj seminarskog rada je istražiti razne algoritme i metode izglađivanja slika da bi se problem oštrih prijelaza i šuma u slikama ublažio.

2. Opis zadatka

Zadan tekst:

Implementirati nekoliko osnovnih algoritama zaglađivanja slike (median filtering, average filtering, gaussian filtering).
Testirati na nekoliko proizvoljno odabranih slika.
Dozvoljeno koristiti gotove biblioteke (openCV)

3. Dobiveni rezultati, kod

Rješenje algoritama je izvedeno u Python programskom jeziku koristeći dostupne biblioteke za obradu slika. Link za kod se nalazi u literaturi.

U radu je korištena *OpenCV* biblioteka za python. *OpenCV* je biblioteka raznih programskih funkcija sa naglaskom na izvođenje u stvarnom vremenu za računalni vid (slike, video i slično). Koristi se i *matplotlib* biblioteka za grafički prikaz slika u trodimenzionalnom prikazu.

3.1. Normalni filter – OpenCV Blur (Average filtering)

Prva i najosnovnija funkcija *OpenCV* biblioteke je *blur()*. Funkcija obavlja konvoluciju nad slikom koristeći neku specifično zadanu jezgru(engl. *Kernel*). Jezgra je matrica sa takvim oblikom da je uvijek jednakih dimenzija u visinu i širinu, te uvijek ima „centralni" član, tj. mora imati neparni broj kao visinu i širinu. Kako jezgra konvoluira tj. prelazi preko slike ona svakom pikselu dodjeljuje novu vrijednost u ovisnosti o jezgri. U slučaju funkcije *blur* ona daje srednju vrijednost svih piksela koji se nalaze u opsegu jezgre (npr. ako je jezgra 5x5 onda će središnji piksel podudarati središnjem članu jezgre, i poprimiti srednju vrijednost svih piksela u 5x5 području okolo tog piksela).

U slučaju jezgre 5x5, jezgra je oblika:

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

U programskom kodu imamo dva načina kako bi to zadali:

- Kreiranjem jezgre i koristeći funkciju *cv2.filter2d()* koja konvoluira neku jezgru nad nekom slikom (isječak koda 3.1.1.)

```
kernel = np.ones((5,5),np.float32)/25
dst = cv2.filter2D(img,-1,kernel)
```

Isječak koda 3.1.1. – primjer 5x5 filtra

- Korištenjem funkcije *cv2.blur()* da bi odmah napravili konvolucijsku operaciju nad slikom (isječak koda 3.1.2.)

```
blur = cv2.blur(img,(5,5))
```

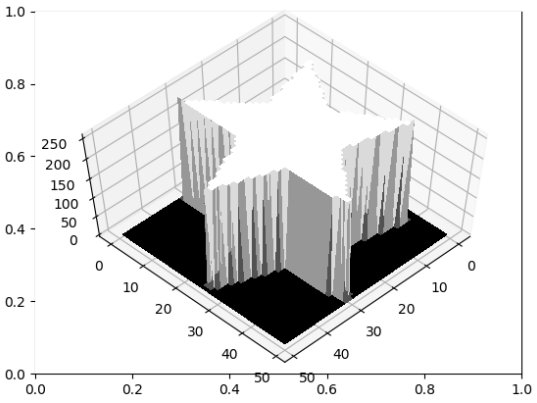
Isječak koda 3.1.2. – primjer 5x5 filtra preko funkcije blur()

Na oba načina su postignuti isti rezultati, razlika je u tome što se kod drugog načina kernel automatski generira iz zadanih parametara.

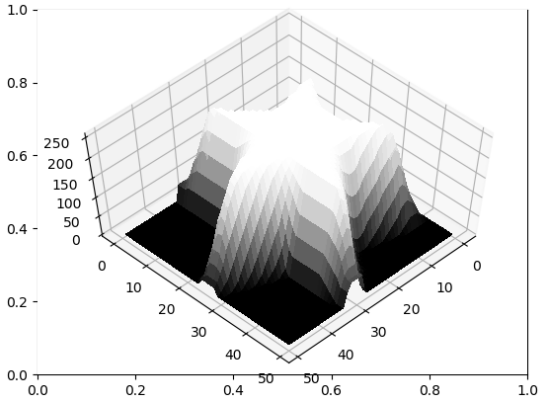
3.1.1. Rezultati nad slikama – oštri prijelazi

Slika 3.1.1.1. – Test slika zvijezda 50x50 piksela

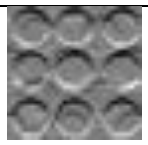
Slika 3.1.1.2. – Slika nakon 5x5 filtra



Slika 3.1.1.3. – Slika zvijezde u 3D prikazu

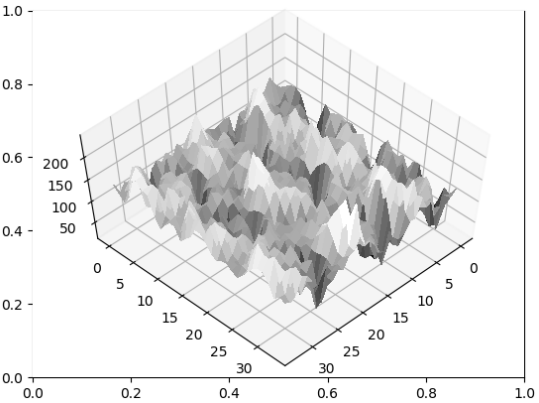


Slika 3.1.1.4. – Slika nakon 5x5 filtra u 3D prikazu

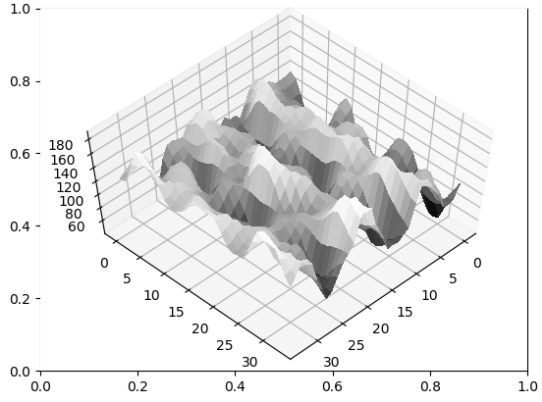


Slika 3.1.1.5. – Test slika reljefa 34x34 piksela

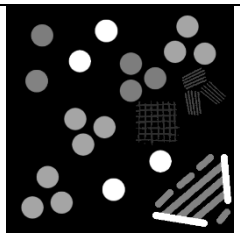
Slika 3.1.1.6. – Slika nakon 3x3 filtra



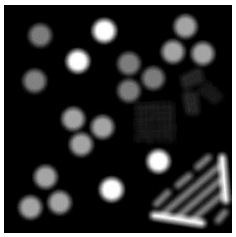
Slika 3.1.1.7. – Slika reljefa u 3D prikazu



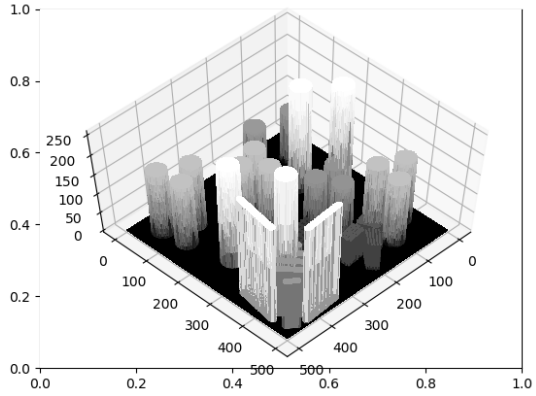
Slika 3.1.1.8. Slika reljefa nakon 3x3 filtra u 3D prikazu



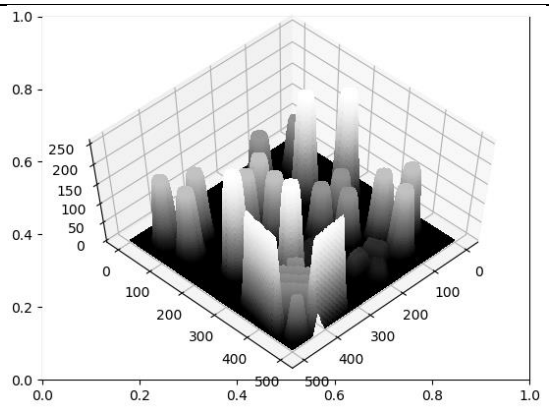
Slika 3.1.1.9. – Test slika nasumičnih nijansi 500x500 piksela



Slika 3.1.1.10. – Slika nakon 15x15 filtra



Slika 3.1.1.11. – Slika u 3D prikazu

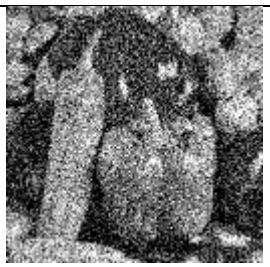


Slika 3.1.1.12. – Slika nakon 15x15 filtra u 3D prikazu

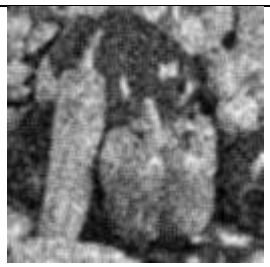
3.1.2. Rezultati nad slikama – šum

Slika 3.1.2.1. – Slika pepper sa uniform noise

Slika 3.1.2.2. – Slika nakon 3x3 filtra



Slika 3.1.2.3. – Slika pepper sa gausovim šumom



Slika 3.1.2.4. – Slika nakon 3x3 filtra

Slika 3.1.2.5. – Slika pepper sa sol i papar šumom

Slika 3.1.2.6. – Slika nakon 3x3 filtra

3.2. Gaussov filter

Funkcija zamućenja koja koristi Gaussovu funkciju. Funkcija radi na sličan način kao i normalno zamućivanje u prethodnoj funkciji ali koristi drugačiju vrstu jezgre. Jezgre su npr. oblika:

- 3x3: $K = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$
- 5x5: $K = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$

Gdje postoje razlike kod parametra *sigma*. Kod sljedećih funkcija taj parametar se može automatski izračunati iz veličine matrice koristeći formulu:

$$\sigma = 0.3 \cdot \big((kernel\ size - 1) \cdot 0.5 - 1\big) + 0.8$$

Te slično kao kod prethodne funkcije postoje dva načina kako odraditi konvoluciju nad slikom:

- Kreiranjem jezgre korištenjem funkcije*cv2.getGaussianKernel()* koju konvoluiramo nad slikom (isječak koda 3.2.1.)

```
ksize=5
sigma=0.3*((ksize-1)*0.5 - 1) + 0.8
kernel = cv2.getGaussianKernel(ksize,sigma)
dst = cv2.filter2D(img,-1,kernel)
```

Isječak koda 3.2.1. – primjer 5x5 filtra gdje se sigma računa iz veličine jezgre

- Direktnim korištenjem funkcije *cv2.GaussianBlur()* (parametar sigma se automaski računa danom formulom ako je dan negativan broj ili nula u parametru)(isječak koda 3.2.2.)

```
blur = cv2.GaussianBlur(img,(5,5),0)
```

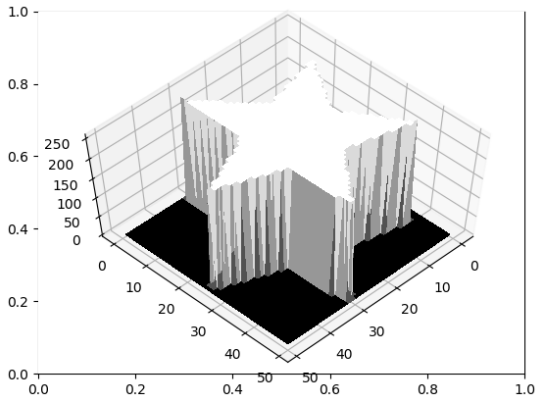
Isječak koda 3.2.2. – primjer 5x5 filtra direktno preko funkcije cv2.GaussianBlur()

Na oba načina je dobiven isti rezultat.

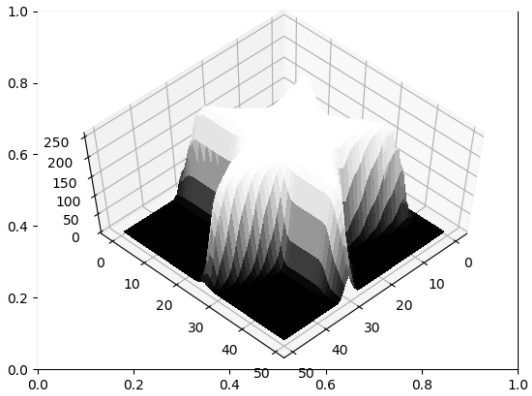
3.2.1. Rezultati nad slikama – oštri prijelazi

Slika 3.2.1.1. – Test slika zvijezda 50x50 piksela

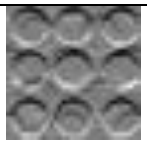
Slika 3.2.1.2. – Slika nakon 5x5 filtra



Slika 3.2.1.3. – Slika zvijezde u 3D prikazu

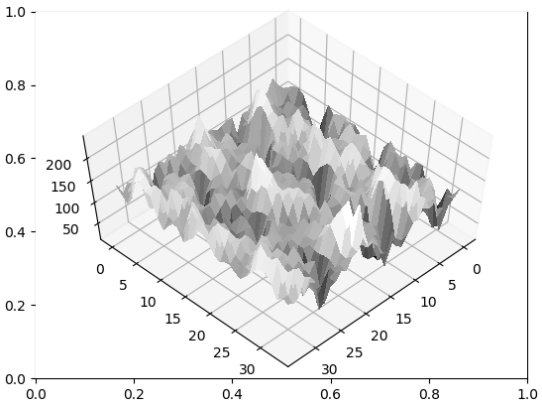


Slika 3.2.1.4. – Slika nakon 5x5 filtra u 3D prikazu

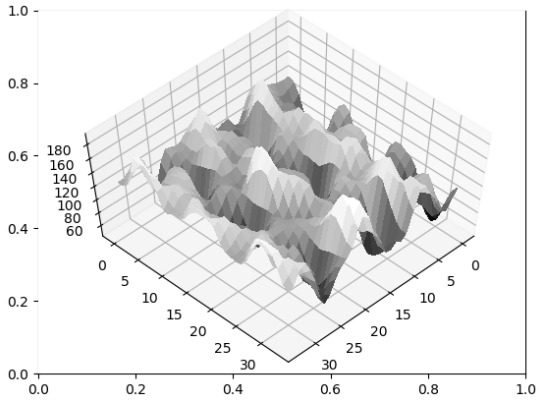


Slika 3.2.1.5. – Test slika reljefa 34x34 piksela

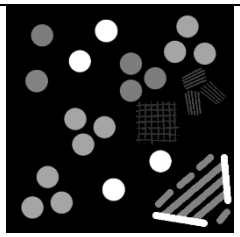
Slika 3.2.1.6. – Slika nakon 3x3 filtra



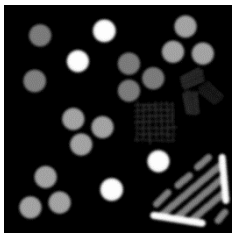
Slika 3.2.1.7. – Slika reljefa u 3D prikazu



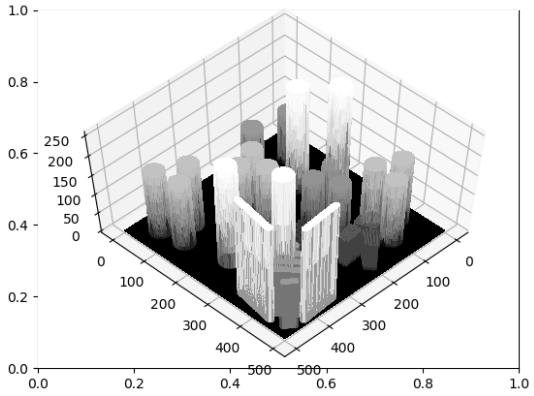
Slika 3.2.1.8. Slika reljefa nakon 3x3 filtra u 3D prikazu



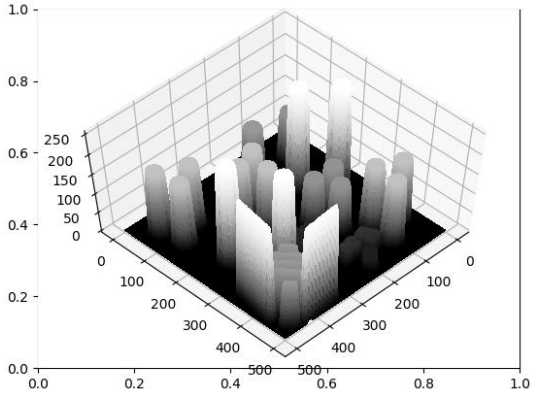
Slika 3.2.1.9. – Test slika nasumičnih nijansi 500x500 piksela



Slika 3.2.1.10. – Slika nakon 15x15 filtra



Slika 3.2.1.11. – Slika u 3D prikazu



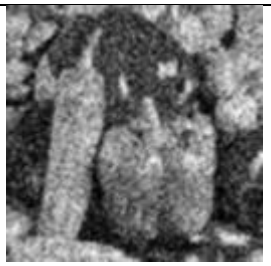
Slika 3.2.1.12. – Slika nakon 15x15 filtra u 3D prikazu

3.2.2. Rezultati nad slikama – šum

Slika 3.2.2.1. – Slika pepper sa uniform noise

Slika 3.2.2.2. – Slika nakon 3x3 filtra

Slika 3.2.2.3. – Slika pepper sa gausovim šumom



Slika 3.2.2.4. – Slika nakon 3x3 filtra

Slika 3.2.2.5. – Slika pepper sa sol i papar šumom

Slika 3.2.2.6. – Slika nakon 3x3 filtra

3.3. Median filter

Median filter je filter koji konvoluira sliku sa jezgrom na takav način da središnji piksel poprimi median veličinu od piksela obuhvaćenih jezgrom. Ovakav filter je efektivan protiv „papar i sol“ šuma, jer kod prethodnih primjera (koji su samo oštri prijelazi bez šuma) neće efektivno djelovati, nego će samo izgladiti nijanse.

Poziva se preko *OpenCV* biblioteke:

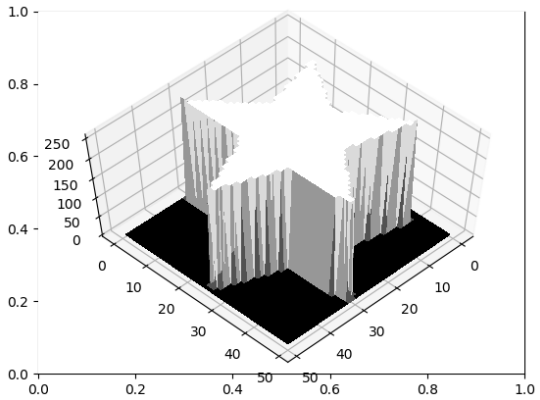
```
median = cv2.medianBlur(img,5)
```

Isječak koda 3.3.1. – primjer 5x5 filtra

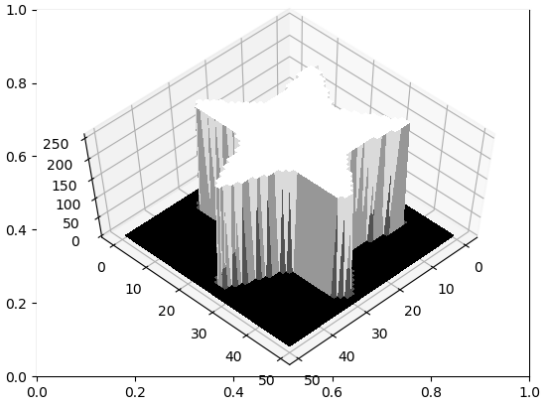
3.3.1. Rezultati nad slikama – oštri prijelazi

Slika 3.2.1.1. – Test slika zvijezda 50x50 piksela

Slika 3.2.1.2. – Slika nakon 5x5 filtra



Slika 3.2.1.3. – Slika zvijezde u 3D prikazu



Slika 3.2.1.4. – Slika nakon 5x5 filtra u 3D prikazu

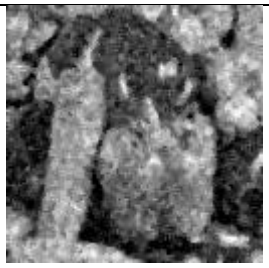
Kao što je prethodno napomenuto, ako se koristi kod primjera za oštar prijelaz dobivaju se neznatni rezultati koji nisu od pomoći (osim ako je cilj smanjiti oštrinu rubova u oblicima, onda ima smisla koristit filter na ovakav način). Potrebno je koristiti slike sa šumom kao primjere.

3.3.2. Rezultati nad slikama – šum

Slika 3.3.2.1. – Slika pepper sa uniform noise

Slika 3.3.2.2. – Slika nakon 3x3 filtra

Slika 3.3.2.3. – Slika pepper sa gausovim šumom



Slika 3.3.2.4. – Slika nakon 3x3 filtra

Slika 3.3.2.5. – Slika pepper sa sol i papar šumom

Slika 3.3.2.6. – Slika nakon 3x3 filtra

Slika 3.3.2.7. – Prikaz učinkovitosti median filtra kod slike sa visokom razinom štete

3.4. Bilateral filter

Bilateralni filter je nelinearni tip filtera koji je dizajniran da očuva rubove slika nad kojima je primijenjen. Kako filter konvolira preko slike, mijenja intenzitet svakog piksela sa težinskom srednjom vrijednošću intenziteta ostalih piksela, gdje se težina bazira na Gaussovoj distribuciji.

Poziva se preko *OpenCV* biblioteke:

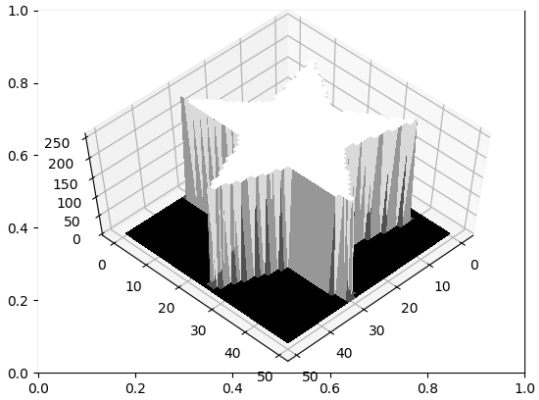
```
cv2.bilateralFilter(img, kernel_size, 90, 90)
```

Isječak koda 3.4.1. – Primjer bilateralnog filtra

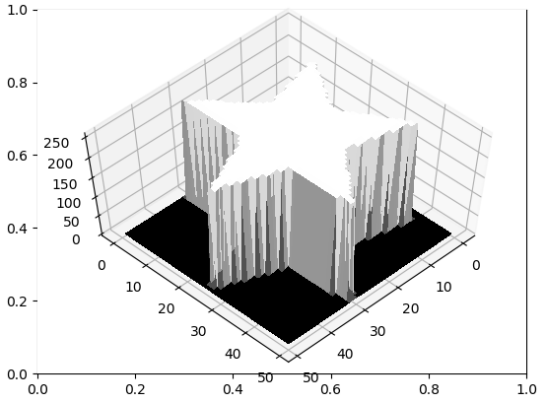
3.4.1. Rezultati nad slikama – oštri prijelazi

Slika 3.2.1.1. – Test slika zvijezda 50x50 piksela

Slika 3.2.1.2. – Slika nakon 5x5 filtra



Slika 3.2.1.3. – Slika zvijezde u 3D prikazu



Slika 3.2.1.4. – Slika nakon 5x5 filtra u 3D prikazu

Slično kao kod median filtera, ovaj filter nema toliki utjecaj na rubne prijelaze tj. neće ugladiti i napraviti prijelaze nad rubovima. Potrebno je pregledati rezultate nad slikama sa šumom.

3.4.2. Rezultati nad slikama – šum

Slika 3.4.2.1. – Slika pepper sa uniform noise

Slika 3.4.2.2. – Slika nakon 5x5 filtra

Slika 3.4.2.3. – Slika pepper sa gausovim šumom

Slika 3.4.2.4. – Slika nakon 5x5 filtra

Slika 3.4.2.5. – Slika pepper sa sol i papar šumom

Slika 3.4.2.6. – Slika nakon 5x5 filtra

Slika 3.4.2.7. – Primjer učinka bilateralnog filtra (desno) nad originalnom slikom (lijevo)

3.5. Korišten kod

Kod je u cijelosti pisan u python programskom jeziku te korištene su sljedeće biblioteke:

numpy	Python biblioteka koja daje podršku za računanje sa velikim višedimenzionalnim matricama i daje veliku kolekciju matematičkih funkcija.
matplotlib	Python biblioteka sa naglaskom na iscrtavanje numeričkih podataka u grafičkim oblicima (npr. razne vrste grafova kao što su histogrami, 3D plotovi, bar plotovi, konturni plotovi, itd.).
cv2	*Open source Computer Vision* je biblioteka programskih funkcija računalnog vida koje su namijenjene stvarno-vremenom izvođenju.

Kod je podijeljen u dvije datoteke: `GUI.py` i `3D.py`.

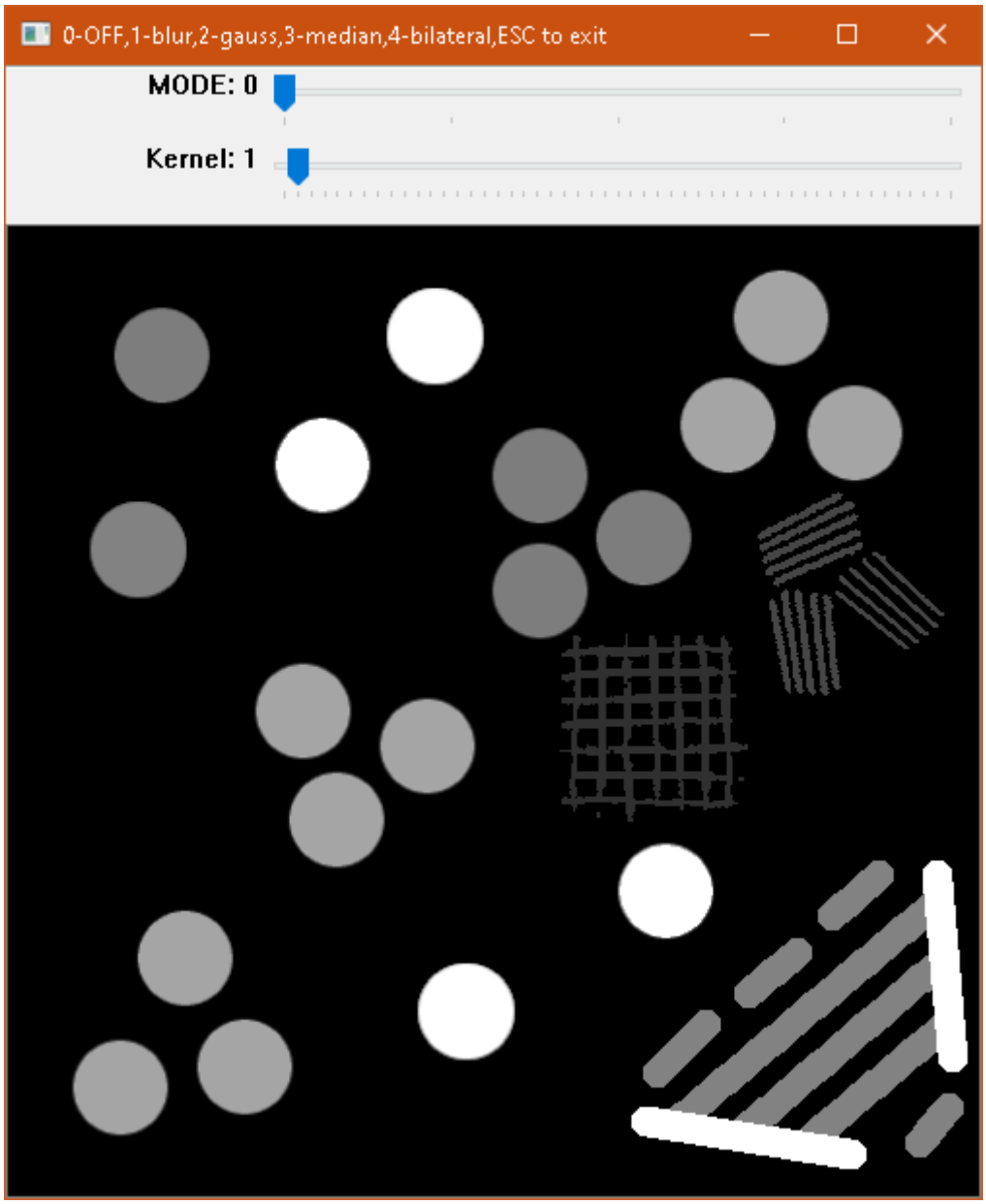
`GUI.py` sadrži kod koji koristi OpenCV biblioteke da bi generiralo jednostavno grafičko sučelje sa kojim se upravlja tip filtra i veličina jezgre pomoću klizača. Ne koristi se vanjska grafička biblioteka nego ugrađene *OpenCV* funkcije (isječak koda 3.5.1.).

```
window_name = "0-OFF,1-blur,2-gauss,3-median,4-bilateral,ESC to exit"
cv2.namedWindow(window_name)

switch = 'MODE'
cv2.createTrackbar(switch, window_name, 0, 4, nothing)

cv2.createTrackbar('Kernel', window_name, 1, 50, nothing)
```

Isječak koda 3.5.1. – Primjer generiranja prozora i klizača za upravljanje filtrima



Slika 3.5.1. – Prikaz grafičkog sučelja sa testnom slikom za prijelaze

Prvi klizač upravlja tipom filtra (0 za bez filtra, 1 za normalno zamućivanje, 2 za gaussovo zamućivanje, 3 za median zamućivanje i 4 za bilateralno zamućivanje). Drugi klizač upravlja veličinom jezgre, problem je u tome što slajderi nemaju kontrolu nad korakom u klizaču, pa se u programskom kodu mora osigurati da se parne vrijednosti ne daju kao veličine jezgre (Isječak koda 3.5.2.):

```
kernel = cv2.getTrackbarPos('Kernel', window_name)
if kernel%2 == 0:
    kernel += 1
```

Isječak koda 3.5.2. – Rješenje problema sa veličinom jezgre

Nakon toga se provjeri koji je filter odabran, te se primijeni i prikaže (Isječak koda 3.5.3.):

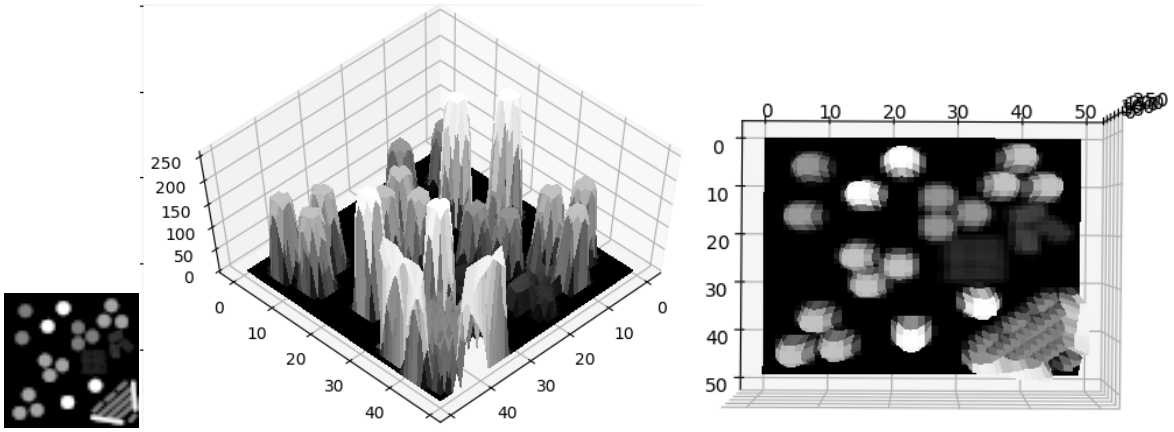
```
s = cv2.getTrackbarPos(switch, window_name)

if s == 0:
    modified_img = img
elif s == 1:
    modified_img = cv2.blur(img, (kernel, kernel))
elif s == 2:
    modified_img = cv2.GaussianBlur(img, (kernel, kernel),0)
    #sigma = 0.3*((ksize-1)*0.5 - 1) + 0.8
elif s == 3:
    modified_img = cv2.medianBlur(img, kernel)
elif s == 4:
    modified_img = cv2.bilateralFilter(img, kernel, 90, 90)

cv2.imshow(window_name, modified_img)
```

Isječak koda 3.5.3. – primjena filtra i prikaz

Druga datoteka, `3D.py`, sadrži kod za prikaz crno bijelih slika u trodimenzionalnom prostoru na takav način da što je veći intenzitet, to je veća vrijednost u trećoj dimenziji.



Slika 3.5.2. – prikaz originalne slike(50x50)(lijevo), slika u izometrijskom prikazu(srednje) te isti prikaz ali ptičji pogled(desno)

Ovaj prikaz je omogućen tako da se *matplotlib* biblioteka proširi sa dodatnim alatom *Axes3D*(isječak koda 3.5.4.):

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

Isječak koda 3.5.4. – Prikaz uključenja dodatne biblioteke za proširenje *matplotlib.pyplot*

To omogućuje korištenje dodatnih alata za trodimenzionalni prikaz podataka. Potrebno je generirati prozor i jedan graf koji koristi trodimenzionalnu projekciju na prozoru da bi mogli prikazati sliku(isječak koda 3.5.5.):

```
figure, ax = plt.subplots()
figure.canvas.set_window_title("3D preview")
ax = figure.add_subplot(111, projection='3d')
```

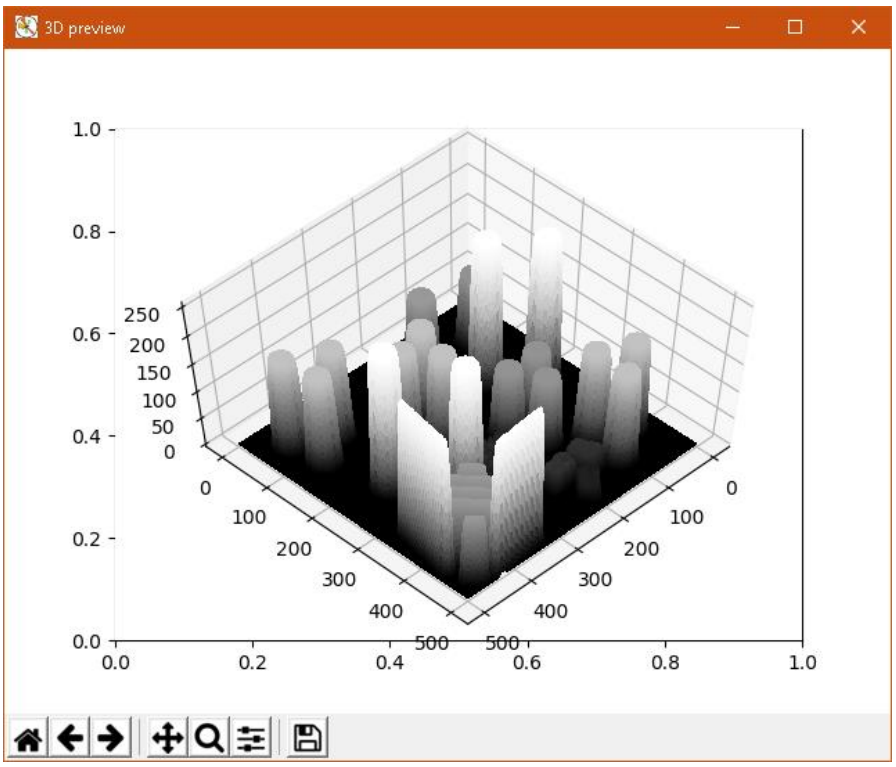
Isječak koda 3.5.5. – Generiranje prozora i grafa

Nakon čega se daje odabir korisniku dali želi visoku rezoluciju trodimenzionalnog prikaza jer kod većih slika (500x500 piksela i veće) treba puno vremena da se prikažu podatci. Nakon odabira se poziva funkcija za crtanje(isječak koda 3.5.6.):

```
xx, yy = np.mgrid[0:height, 0:width]
stride_answer = input("high resolution of strides in 3D? (high means a lot
of lag but more detail, no is width/100,height/100 for strides)(Y/N): ")
if(stride_answer == 'Y' or stride_answer == 'y'):
    ax.plot_surface(xx, yy, img, cmap='Greys_r',rstride=1, cstride=1, antial
iased=False, linewidth=0)
else:
    ax.plot_surface(xx, yy, img, cmap='viridis',rstride=round(width/100), cs
tride=round(height/100), antialiased=False, linewidth=0)
ax.view_init(60, 45)
plt.show()
```

Isječak koda 3.5.6. – Kod za crtanje trodimenzionalnog prikaza

Što nam kao rezultat daje *matplotlib* prozor sa grafičkim prikazom kojeg možemo manipulirati mišem(možemo ga okretati, te koristeći dugmad na prozoru možemo povećavati ili smanjivati prikaz kao na slici 3.5.3.), ili u programu sa *ax.view_init()* možemo mijenjati kut gore/dolje(engl. *elevation*) i lijevo/desno (engl. *azimuth*), npr. ako želimo 360° snimku objekta u kodu navedemo da nam mijenja azimuth za 1° u petlji.



Slika 3.5.3. – Prozor za trodimenzionalni prikaz

4. Kratki zaključak

Kod algoritama zaglađivanja slika ne postoji najoptimalniji ili najbolji algoritam za sve situacije, jer konačni rezultat uvelike ovisi o stanju izvorne slike (kakav šum ima, kolika je kvaliteta detalja na slici) te o odabranim parametrima filtera (npr. prevelika jezgra uzrokuje bolje uklananje šuma i veće prijelaze ali se i gubi više detalja u slici, tj. smanjuju se dostupne informacije).

Za uklananje oštrih prijelaza koriste se „average“ tj normalni filter zaglađivanja i Gaussov filter zaglađivanja. Kod slika 3.1.1.4. za normalni filter i slike 3.2.1.4. za Gaussov filter za trodimenzionalni prikaz se vidi razlika u konturi spusta iz visokog intenziteta u niski. Kod normalnog je linearan spust dok kod Gaussovog poprima oblik Gaussove distribucije. Ako je potrebno za ovakav problem koristiti manju jezgru jer je npr. slika manjih dimenzija onda je normalan filter bolji izbor za manju jezgru jer ima manju mogućnost da će zapeti negdje sa oštrim prijelazom kod manje slike, dok za veće slike je pogodniji Gaussov filter jer se može korisitit manja jezgra nego normalni filter i dalo bi pogodniji prijelaz bez problema visoke strmine za diferencijske funkcije.

Za uklananje šuma se može koristiti bilokoji navedeni filter u radu,svi imaju svoje mane i prednosti. Normalni i Gaussov filter ovise o veličini jezgre i Gaussov dodatno ovisi o parametru sigma koji mu se zadaje, te ovi filtri ne mogu sačuvati rubove originalnih slika.

Median filter radi tako da odabere median vrijednost piksela oko odabranog piksela (ovisno o veličine jezgre). Nelinearan je zato što median znači da odabire već neku postojeću vrijednost a ne preko računalnog algoritma, te zato što radi na takav način može sačuvati rubove originalne slike te vrlo je učinkovit protiv sol i papar šuma jer pogrešne vrijednosti ne utječu na konačnu vrijednost piksela(slika 3.3.2.7.).

Bilateralni filter je nelinearan filter, slično kao i median čuva rubove, te ima algoritam u sebi koji spriječava izglađivanje preko rubova, dok normalno filtrira unutar rubova. Na ovaj način filter može sačuvati oštre rubove (kao što je prikazano kod test slike zvijezde ili kod slike planina 3.4.2.7.).

5. Literatura

Link za korišten kod u projektu:
https://github.com/ATufekovic/OSIRV_zavrsni_projekt_Anto_Tufekovic

https://docs.opencv.org/master/d4/d86/group__imgproc__filter.html

https://en.wikipedia.org/wiki/Box_blur

https://en.wikipedia.org/wiki/Gaussian_filter

https://en.wikipedia.org/wiki/Median_filter

Slika korištena kod median filtera (Slika 3.3.2.7.):

By The original uploader was Anton at German Wikipedia. - Originally from de.wikipedia; description page is/was here., CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=2544834

https://en.wikipedia.org/wiki/Bilateral_filter

Slika korištena kod bilateralnog filtera(Slika 3.4.2.7.):

By U.S. National Park Service / Phrood - http://commons.wikimedia.org/wiki/File:MinaretLake.jpg, Public Domain, https://commons.wikimedia.org/w/index.php?curid=27849862