SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I
INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij računarstva

Markov random field model za uklanjanje šuma i segmentaciju slika

Obrada slike i računalni vid

Patrik Juzbašić

Osijek, 2023.

Sadržaj

1. UVOD

U današnjem digitalnom dobu, obrada slika je postala neizostavan dio mnogih područja kao što su medicina, sigurnost, prepoznavanje uzoraka i mnogi drugi. Jedan od ključnih izazova u obradi slika je uklanjanje šuma i segmentacija slika, što ima izravan utjecaj na kvalitetu rezultata i interpretaciju slika. Uklanjanje šuma se odnosi na proces u kojem se pokušava eliminirati šum koji je prisutan u digitalnim slikama. Šum može biti uzrokovan raznim faktorima kao što su senzorske pogreške, loša kvaliteta snimanja ili kompresija slike. S druge strane, segmentacija slika je postupak kojim se slika dijeli na različite regije ili objekte kako bi se olakšala analiza i interpretacija. U ovom radu fokusiramo se na primjenu Markov random field (MRF) modela za uklanjanje šuma i segmentaciju slika. MRF je probabilistički model koji koristi koncept Markovljevih svojstava kako bi modelirao prostornu ovisnost piksela u slici. MRF model se temelji na teoriji grafova i Bayesovom teoremu te ima široku primjenu u različitim područjima obrade slika.

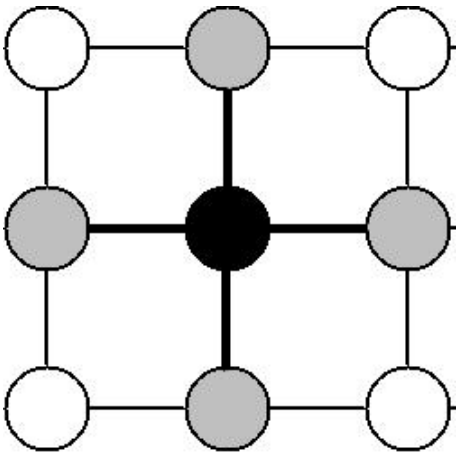
1.1. Zadatak rada

Zadatak ovog rada je razviti sustav za uklanjanje šuma i segmentaciju slika koristeći Markov random field (MRF) model. To će uključivati objašnjenje teorijskih osnova probabilističkih modela i MRF-a, implementaciju algoritma te evaluaciju rezultata kroz mjere poput PSNR-a i MSE-a.

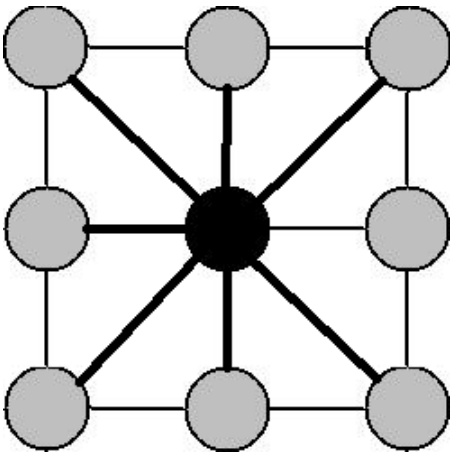
2. PREGLED PODRUČJA

Područje uklanjanja šuma i segmentacije slika ima veliku važnost u digitalnoj obradi slika. Postoje razni pristupi koji se koriste za ove zadatke, uključujući statističke metode, filtriranje u domeni frekvencija i upotrebu MRF modela. MRF modeli su postali popularni zbog svoje sposobnosti modeliranja prostornih ovisnosti piksela i dobrih performansi u različitim zadacima obrade slika.

MRF je probabilistički model koji se koristi za modeliranje prostorne ovisnosti piksela u slici. Temelji se na teoriji grafova i Bayesovom teoremu. MRF model pretpostavlja da je svaki piksel u slici uvjetno nezavisan o svim ostalim pikselima, osim onih koji su mu susjedi. Ova pretpostavka omogućuje modeliranje kontekstualnih informacija i ovisnosti između piksela, što je ključno za uklanjanje šuma i segmentaciju slika. Prikaz susjeda pikselu se nalazi na slikama 2.1 i 2.2 i predstavljaju susjede piksela ako se uzmu u obzir i diagonalni, tada piksel ima osam susjede slika 2.2, ako ako se oni zanemare tada ima 4 susjeda slika 2.1. MRF se temelji na izračunu funkcije potencijala te energijske funkcije.



SL. 2.1: *Četiri susjeda*



SL. 2.2: *Osam susjeda*

2.1. Korištene tehnologija

U ovom radu je korišten programski jezik Python kao osnovna tehnologija za implementaciju algoritama i sustava za uklanjanje šuma i segmentaciju slika. Python je popularan jezik u području znanstvenog računanja i obrade slika zbog svoje jednostavnosti, ekspresivnosti i bogatih biblioteka. Jedna od ključnih biblioteka korištenih u ovom radu je OpenCV (Open Source Computer Vision Library). OpenCV je otvorena biblioteka specijalizirana za obradu slika i kompjutorsko gledanje. Pruža bogat skup algoritama i funkcionalnosti za manipulaciju slikama, uključujući učitavanje i spremanje slika, filtriranje, segmentaciju, praćenje objekata itd. OpenCV je vrlo popularan alat u području računalnog vida i pruža efikasne implementacije mnogih algoritama za obradu slika. Također, biblioteka numpy se često koristi za numeričke operacije i manipulaciju podacima. Numpy pruža efikasne i optimizirane funkcije za rad s višedimenzionalnim poljima podataka, što je vrlo korisno u radu s matricama i slikama. Uz to, za implementaciju Markovljevog slučajnog polja (MRF) modela i probabilističkih modela, mogu se koristiti razne biblioteke i okviri poput PyMC3, scikit-learn ili TensorFlow Probability. Ove biblioteke pružaju alate za izgradnju, treniranje i primjenu probabilističkih modela, uključujući MRF modele, te omogućuju manipulaciju vjerojatnostima, optimizaciju i inferenciju.

MSE (Mean Squared Error) je mjera koja kvantificira prosječnu kvadratnu razliku između intenziteta piksela originalne slike i rekonstruirane slike. Izračunava se tako što se za svaki piksel izračuna kvadratna razlika između intenziteta piksela u originalnoj i rekonstruiranoj slici, a zatim se te razlike srednje vrijednosti. PSNR (Peak Signal-to-Noise Ratio) je mjera koja se temelji na MSE-u, ali se izražava u logaritamskoj skali i koristi se za kvantificiranje omjera između maksimalne snage signala i kvadrata srednjekvadratne pogreške. PSNR je definiran kao 2.4, gdje je "max" maksimalna moguća vrijednost piksela (npr. 255 za 8-bitne slike). Niži MSE i veći PSNR rezultati ukazuju na bolju rekonstrukciju slike.

```python
from PIL import Image
import random
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from math import log10, sqrt
import cv2
```

SL. 2.3: *Korištene biblioteke*

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

SL. 2.4: *PSNR*

3. PROGRAMSKO RJEŠENJE

Prvi korak je bio dodavanje Gausovog šuma na slike koje ćemo koristit za segemntaciju.

```
def add_gaussian_noise(load_path, save_path, mean, var):
    original_img = cv2.imread(load_path)
    noisy_img = skimage.util.random_noise(original_img, mode='gaussian', mean=mean, var=var)
    noisy_img = np.array(255*noisy_img, dtype = 'uint8')
    cv2.imwrite(save_path, noisy_img)
    plt.imshow(mpimg.imread(save_path))
    plt.title("Image with "+str(var)+ " noise variance")
    plt.show()
    return
```

SL. 3.5: *Dodavanje šuma*

Zatim se provodi postupak prilagodbe normalne distribucije za svaku klasu piksela u skupu za treniranje.

```
def fit_normal_dist_per_class(original_image_filepath, train_image_filepath, other_pixels_color):
    image_per_label_pixels = set()
    orig_im = Image.open(original_image_filepath).convert('LA')
    orig_pixels = orig_im.load()
    width, height = orig_im.size
    i = 0
    j = 0
    while (i < width):
        while (j < height):
            pix = orig_pixels[i, j][0]
            if (pix != other_pixels_color):
                image_per_label_pixels.add((i, j))
            j += 1
        i += 1
        j = 0
    train_im = Image.open(train_image_filepath).convert('LA')
    train_pixels = train_im.load()
    data = [train_pixels[i, j][0] for (i, j) in image_per_label_pixels]
    mean, std = norm.fit(data)
    return mean, std, image_per_label_pixels
```

SL. 3.6: *Prilagodba normalne distribucije*

Zatim se primjenjuje MRF model koji koristi lokalne susjede piksela i energijske funkcije kako bi odredio najvjerojatniju klasu za svaki piksel. Također, kod uključuje funkciju report_segment_image_mrf koja pokreće proces segmentacije slike. Ona prvo prilagođava distribucije po klasama, zatim pokreće MRF algoritam i na kraju sprema rezultate segmentacije. Slikama 3.8 i 3.9 prikazane su funkcije za dohvačanje energije slike.

```python
def report_segment_image_mrf(base_filepath,base_path_output, test_image_filename, labeled_image_filenme, noisy_train_image_filename,
                             class_original_image_filenames, other_pixels_colors,
                             beta=5, t=100, neighbor_count=4,
                             true_label_prob=0, t_ratio=0.97):
    true_label_in_init_count = 0
    all_classes_image_per_label_pixels, class_means, class_stds = fit_class_dists(class_original_image_filenames,
                                                                                  noisy_train_image_filename,
                                                                                  other_pixels_colors)


    _, _, pixel_class_indexes = segment_image(base_filepath,
                                              test_image_filename, all_classes_image_per_label_pixels,
                                              class_means, class_stds)
    fixed_pixel_indexes = set()

    for i in range(len(pixel_class_indexes)):
        for j in range(len(pixel_class_indexes[0])):
            if random.uniform(0, 1) > true_label_prob:
                pixel_class_indexes[i][j] = random.randint(0, len(class_means) - 1)
            else:
                for idx, all_classes_image_per_label_pixel in enumerate(all_classes_image_per_label_pixels):
                    if (i, j) in all_classes_image_per_label_pixel:
                        pixel_class_indexes[i][j] = idx
                        true_label_in_init_count += 1
                        break

    image_segmentation_mrf = CustomImageSegmentationMRF()
    new_pixel_class_indexes = image_segmentation_mrf.start(500000, pixel_class_indexes,
                                                           base_path + test_image_filename, class_means, class_stds,
                                                           fixed_pixel_indexes, beta, t, neighbor_count, t_ratio)

    width = len(new_pixel_class_indexes)
    height = len(new_pixel_class_indexes[0])
    saving_im = Image.new('LA', (width, height))
    im = Image.open(base_filepath + test_image_filename).convert('LA')
    pixels = im.load()
    all_pixels_count = 0
    all_trues_count = 0
    for i in range(width):
        for j in range(height):
            saving_im.putpixel((i, j), (int(class_means[new_pixel_class_indexes[i][j]]), pixels[i, j][1]))
            all_pixels_count += 1
            if (i, j) in all_classes_image_per_label_pixels[new_pixel_class_indexes[i][j]]:
                all_trues_count += 1
    saving_im.save(base_path_output + labeled_image_filenme, 'png')

    print("accuracy: " + str(float(all_trues_count) / float(all_pixels_count)))
    return float(all_trues_count) / float(all_pixels_count), saving_im, true_label_in_init_count
```

SL. 3.7: *Funkcija za pokretanje segmentacije*

```python
def get_total_energy(self):
    e = 0
    self.j_max = len(self.state[0]) - 1
    self.i_max = len(self.state) - 1
    for i in range(len(self.state)):
        for j in range(len(self.state[0])):
            e += self.get_neighbors_sum_energy_for_total_energy_computing(i, j,
                                                                          self.neighbor_count) + self.get_pix_energy_just_for_intensity(
                i, j)
    return e
```

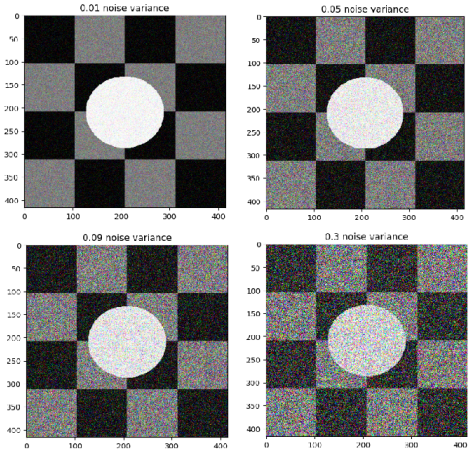
SL. 3.8: *Funkcija za dohvačanje energije slike*

```python
def get_neighbors_sum_energy_for_total_energy_computing(self, i, j, neighbor_count):
    e = 0
    if i != 0:
        e = e - 1 if self.state[i][j] == self.state[i - 1][j] else e + 1

    if j != 0:
        e = e - 1 if self.state[i][j] == self.state[i][j - 1] else e + 1

    if neighbor_count == 8:
        if i != 0 and j != 0:
            e = e - 1 if self.state[i][j] == self.state[i - 1][j - 1] else e + 1

        if i != 0 and j != self.j_max:
            e = e - 1 if self.state[i][j] == self.state[i - 1][j + 1] else e + 1

    return e * self.beta
```

SL. 3.9: *Funkcija za dohvačanje energije susjeda*

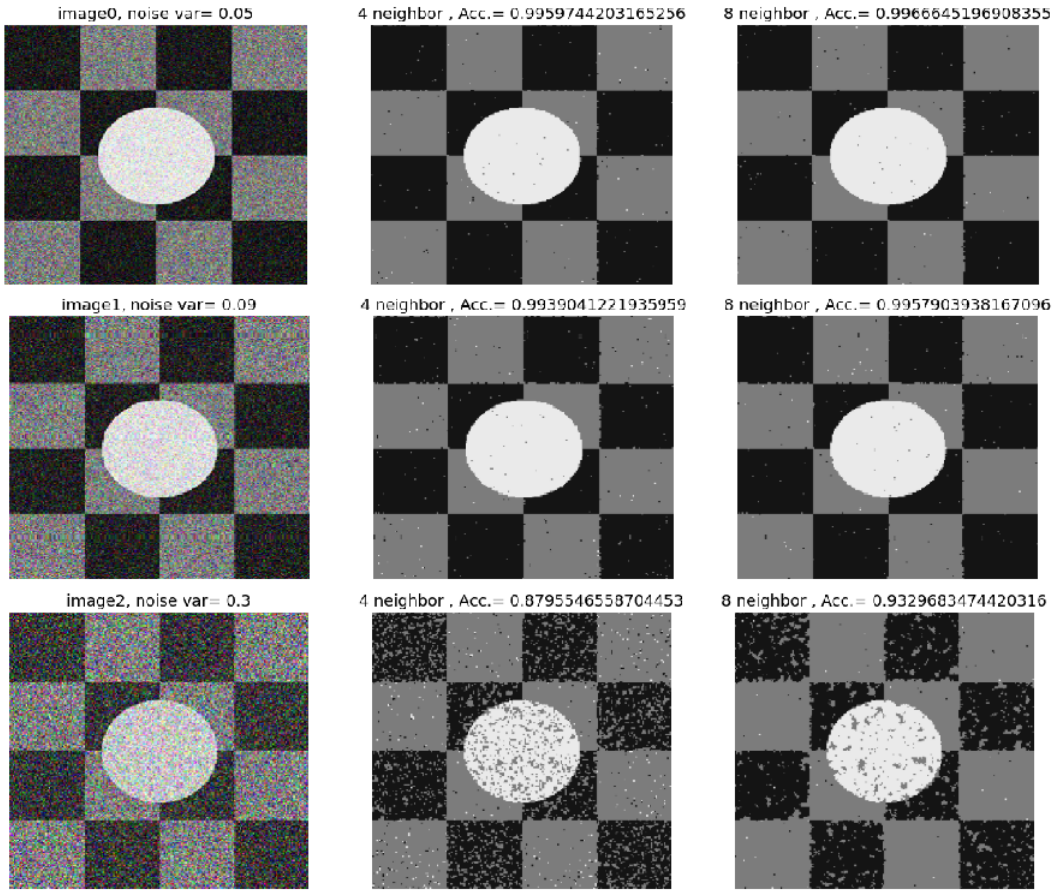
3.1. Rezultati modela

Izgled Gaussova šum na slici sa varijancama šuma u rasponu od 0.01 do 0.3 predstavljen je slikom 3.10.

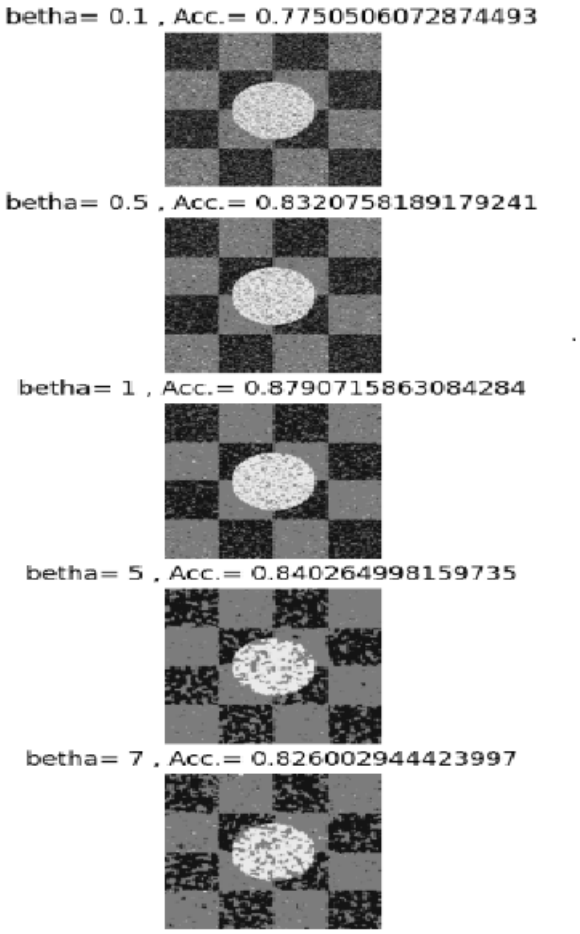


SL. 3.10: *Slike sa šumom*

Rezultati testa razlike u paramterima modela su predstavljeni slikama 3.11 i 3.12, testirani su parametar broj susjeda i parametar beta. Tražena je što veća preciznost.



SL. 3.11: *Razlike u paramteru modela broj susjeda*

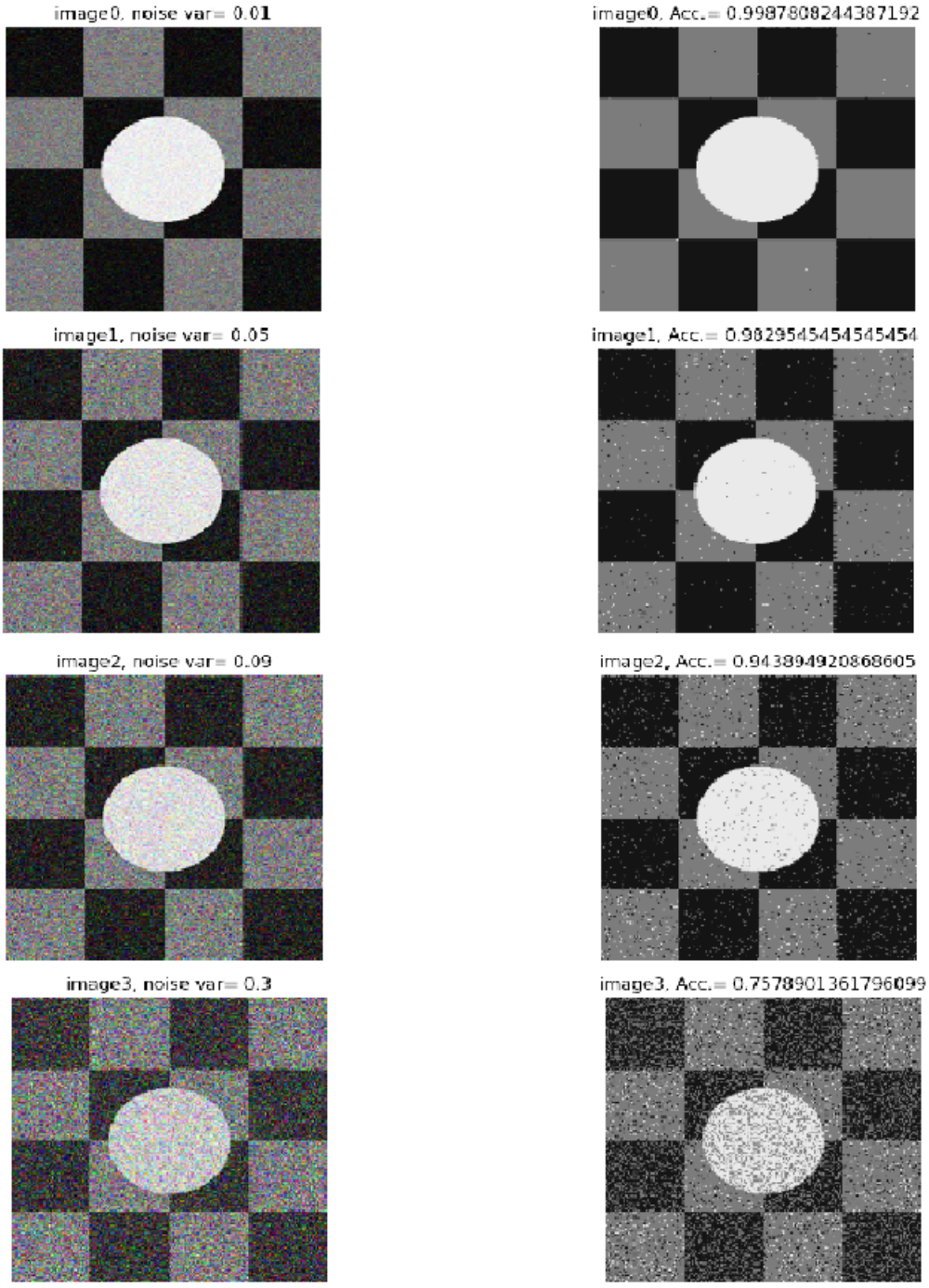


SL. 3.12: *Razlike u paramteru modela beta*

Slikom 3.13 je predstavljeno konačno rješenje problema segmentacije. Očekivano najveću preciznost ima slika s najmanjom varijancom šuma. A tablicom 3.1 MSE i PSNR gdje možemo zaključiti da je za varijance 0.01, 0.05 i 0.09 mala razlika u kvaliteti slike jer se PSNR razlikuje u decimalama. Razmatranjem slikovnog rješenja vidimo da kod slika sa varijancama šuma 0.01 i 0.05 gotov nema vidljivog šuma, dok se šum na slici s 0.3 varijancom nije maknuo te je lako uočljiv.

Varijanca šuma	MSE	PSNR
0.01	28.3438	95.2121
0.05	28.3498	95.0818
0.09	28.3512	95.0504
3	28.9989	81.8821

Tab. 3.1: *MSE I PSNR za obrađene slike.*



SL. 3.13: *Segmentirane slike metodom MRF-a*

4. ZAKLJUČAK

U ovom radu razvijen je sustav za uklanjanje šuma i segmentaciju slika koristeći Markov random field model. Rezultati evaluacije sustava pokazuju da je MRF model učinkovit u uklanjanju šuma i segmentaciji slika. PSNR i MSE metrike potvrđuju visoku kvalitetu dobivenih rezultata. Ovakvi sustavi mogu pružiti korisne alate u područjima kao što su medicinska dijagnostika, sigurnost i prepoznavanje uzoraka.

LITERATURA

[1] C. Li, M. Wand, Markov Random Field Models in Computer Vision. Foundations and Trends in Computer Graphics and Vision

[2] R. C. Gonzalez, R. E. Woods, Digital Image Processing (3rd ed.). Prentice Hall

PRILOG

Kod i rješenja se nalaze na Google Colab-u te su spremljeni na github na sljedećem linku: https://github.com/Patra3007/OSiRV/blob/main/MRF.ipynb. Kod je već pokrenut sa ulaznim podacima i rezultati su vidljivi bez potrebe ponovnog pokretanja. Ukoliko želite ponovno pokrenuti kod potreno je u Notebook-u napraviti dvije mape: prvu nazvati input te u nju postaviti slike koje se nalaze na github linku u mapi 'input', a drugu nazvati output i ostaviti praznu.