# Digitalna obrada slike u Pythonu

**Poljak, Denis**

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni preddiplomski studij matematike

**Denis Poljak**

# Digitalna obrada slike u Pythonu

Završni rad

Osijek, 2020.

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni preddiplomski studij matematike

**Denis Poljak**

# Digitalna obrada slike u Pythonu

Završni rad

Mentor: izv. prof. dr. sc. Domagoj Matijević
Komentor: mag. phys. Ana Petrinec

Osijek, 2020.

# Sažetak

Tema ovog završnog rada je digitalna obrada slike u programskom jeziku Python, a bit će prikazane osnovne informacije o programskom jeziku i njegovim modulima koji su potrebni u obradi i analizi slike. U ovom radu promatrat će se svojstva slike i njezine bitne karakteristike, a potom odrediti na koji način definirati, učitati i analizirati ili modificirati sliku i u konačnici spremiti modificiranu sliku u Pythonu. S obzirom da digitalna obrada slike obuhvaća široka i raznolika polja, cilj ovog rada je i detaljnije objasniti iz kojeg razloga se koriste različiti filteri kao i primjenu različitih algoritama kako bi poboljšali slikovne informacije slike u svrhu znanstvenih i drugih istraživanja. Rad je rađen u suradnji s tvrtkom Orqa, mentoricom Anom Petrinec.



# Ključne riječi

Python, Greyscale, Histogram, Median filter, Max filter, Min filter, Gama korekcija, Segmentacija, Otsu's metoda, Thresholding

# Image processing in Python

## Summary

The subject of this final work is digital image processing in the programming language Python, basic information about the programming language and its modules required in image processing and analysis will be presented. In this paper, the properties of the image and its essential characteristics will be observed, then it will be determined how to define, load and analyze or modify the image and in the end save the modified image in Python. Since image processing covers a wide, diverse field, the aim of this paper is to explain in detail why different filters are used as well as the application of different algorithms to improve image information of the image for scientific and other research.The work was done in collaboration with the company Orqa, mentor Ana Petrinec.

## Key words

Python, Greyscale, Histogram, Median filter, Max filter, Min filter, Gamma correction, Segmentation, Otsu's method, Thresholding

# Sadržaj

# Uvod

Python je popularan programski jezik visoke razine. Tvorac ovog programskog jezika je Nizozemac Guido van Rossum koji ga je počeo razvijati 1990. godine.

Python podnosi razne programske zadatke kao što su numeričko računanje, web razvoj, programiranje baza podataka, mrežno programiranje i u konačnici obrada digitalne slike koja će biti i predmetom ovoga završnog rada. Najveća prednost Python-a u odnosu na druge programske jezike je u tome što je besplatan i dostupan u svim operativnim sustavima kao što su Windows, Mac ili Linux.

Sintaktički je jednostavniji od C/C++ što znači da je visoko čitljiv i lak za uklanjanje pogrešaka. Dolazi s različitim modulima koji su standardni ili se mogu instalirati na postojeću Python instalaciju, odnosno Python je „open source“ gdje je dozvoljena slobodna izmjena i sam razvoj Python-a.

Moduli mogu obavljati različite zadatke poput čitanja i pisanja različitih datoteka, vizualizacija podataka ili slika. Koliko je Python značajan stvarna je činjenica da je u njemu napisan YouTube, najveći svjetski videoservis, veliki dio Googleova pretraživača te Instagram.

# 1 Uvod u Python

## 1.1 Python Moduli

Moduli omogućavaju logično organiziranje Python koda. Grupiranje srodnog koda u modul olakšava razumljivost i upotrebu koda. Jedni od najbitnijih modula koji su potrebni u obradi digitalne slike u Pythonu su Scikit-image, SciPy i OpenCV.

### 1.1.1 Scikit-image

Sckit-image, ranije sckits.image je modul za obradu slike uključuje algoritme za segmentaciju, geometrijske transformacije, manipulacije s bojama, analizu filtriranja, morfologiju i druge. Implementiran je za interakciju s numeričkim i znanstvenim bibliotekama Python-a NumPy i SciPy.

**Primjer 1.** *Pripadni Python kod:*

```
1 import skimage
```

### 1.1.2 SciPy

SciPy se prvenstveno koristi za matematičke i znanstvene proračune, ali također sadrži algoritme za manipulaciju sa slikama poput mjerenja predmeta na slikama, linearno i nelinearno filtriranje, prilagođavanje interpolaciji, filtriranje i razne druge efekte. Osim toga moguće je provođenje i segmentacija nad slikama.

**Primjer 2.** *Pripadni Python kod:*

```
1 import scipy
```

### 1.1.3 OpenCV

OpenCv je usmjeren na obradu slike, prepoznavanje lica, otkrivanje objekata i slično. Napisan je u C++, ali dolazi i sa Pythonom u tandemu s NumPy, SciPy i Matplotlib. Jedan od ciljeva OpenCV-a je stvaranje jednostavne infrastrukture koja omogućava brzu izradu složenih vizijskih aplikacija.

**Primjer 3.** *Pripadni Python kod:*

```
1 import cv2
```

# 2 Slika i svojstva slike

U ovom poglavlju riječ je o vrsti slika, obradi slika i strukturiranju slika u Pythonu i na koji način se definirana i učitana slika prikazuje te u konačnici nakon postupka analiziranja ili modifikacija sliku sprema.

## 2.1 Slika u Pythonu

Krenimo od toga kako je slika definirati i na koji način je reprezentirana kao struktura u Pythonu. Slika je definirana kao funkcija $f(x, y)$ gdje su x i y prostorne kordinate, a amplituda funkcije na bilo kojem paru kordinata $(x, y)$ naziva se intenzitetom slike na toj točki odnosno tu točku nazivamo piksel slike.

Piksel (eng. pixel) izvedenica je od eng. "picture element" što znači element slike. Kombinacijom piksela stvaramo sliku. U tom smislu slika se može definirati kao dvodimenzionalni niz posebno uređenim u redove i stupce. Odnosno slike su reprezentirane u Pythonu kao NumPy nizovi.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & \dots & f(0,n-1) \\ f(1,0) & f(1,1) & f(1,2) & \dots & f(1,n-1) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ f(m-1,0) & f(m-1,1) & f(m-1,2) & \dots & f(m-1,n-1) \end{bmatrix}$$

Desna strana jednakosti je digitalna slika po definiciji. Svaki element ove matrice je piksel slike. Gledano iz daljine, čini se da se ti pikseli spajaju u jednu sliku, što je slika koju vidimo.

Digitalne slike koriste neki model boja za stvaranje širokog raspona boja iz malog niza osnovnih boja. Iako postoji nekoliko različitih modela boja koji se koriste za slike, najčešće se pojavljuje model RGB (crveni, zeleni, plavi).

RGB model je model aditivnih boja, što znači da se primarne boje miješaju zajedno u druge boje. Svaku primarnu boju često nazivamo kanalom. Svaka određena boja u RGB modelu može se izraziti trostrukim cijelim brojevima u [0, 255] rasponu, koji predstavljaju crveni, zeleni i plavi kanal. Veći broj u kanalu znači da je prisutno više od te primarne boje.

## 2.2 Čitanje prikazivanje i spremanje slika

U programu Python koristiti ćemo skimage modul koji pruža funkcije za jednostavnije čitanje, prikazivanje i spremanje slika. Podržani su svi popularni formati slika poput BMP,PNG,JPG i TIFF zajedno s još nekoliko drugih formata.

**Primjer 4.** *Sada učitajmo sliku i općenito promatramo njena različita svojstva.*

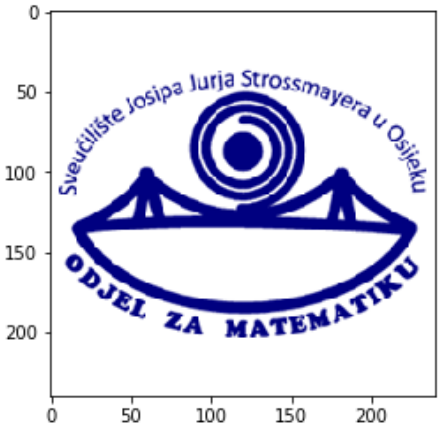
```
1    from skimage import io
2    image = io.imread("slika.jpg")
```

Prvo je uključen io modul skimage kako bi se mogle čitati i prikazati slike. Nastavno se koristi funkcija io.imread () za čitanje JPEG slike pod nazivom slika.jpg. Skimage čita sliku, pretvara je iz JPEG-a u NumPy niz i vraća niz; niz se sprema u varijablu imenom image.

**Primjer 5.** *Slijedom gore učitanu sliku prikazujemo:*

```
1    from skimage import io
2    import matplotlib.pyplot as plt
3
4    image = io.imread("slika.jpg")
5    plt.imshow(image)
```

Nakon što je u programu prikazana slika, učitava se modul pyplot iz biblioteke matplotlib kao plt i prikazuje se slika pomoću plt.imshow().




Slika 1: Slika loga Odjela za matematiku

Neka od osnovnih svojstava slike:

```
from skimage import io
import matplotlib.pyplot as plt

image = io.imread("slika.jpg")

print('Oblik slike : {}'.format(image.shape))
print('Visina slike {}'.format(image.shape[0]))
print('Širina slike {}'.format(image.shape[1]))
print('Dimenzija slike {}'.format(image.ndim))
```

Oblik slike: (240, 240, 3)
Visina slike 240
Širina slike 240
Dimenzija slike 3

Na kraju preostalo je pokazati način na koji se slika sprema. Većinom nije slučaj da se sprema originalna učitana sliku nego se izrađuju određene modifikacije na slici poput promjene veličine slike, formata slike i slično. Slici 1. mijenja se dimenzija slike odnosno oblik te se tom radnjom dobiva nova slika (Slika 2).

```
from skimage import io
import matplotlib.pyplot as plt
from skimage.transform import resize

image = io.imread("slika.jpg")
resize_image = resize(image, (300, 400))
plt.imshow(resize_image)
```


Slika 2: Slika promijenjenog oblika

## 2.3 Greyscale

Crno-bijele slike spremaju se u dvodimenzionalne nizove. Postoje dvije vrste crno-bijelih slika.

- Greyscale: Raspon nijansi sive: 0-255
- Binary: Pikseli su crni ili bijeli: 0 ili 255

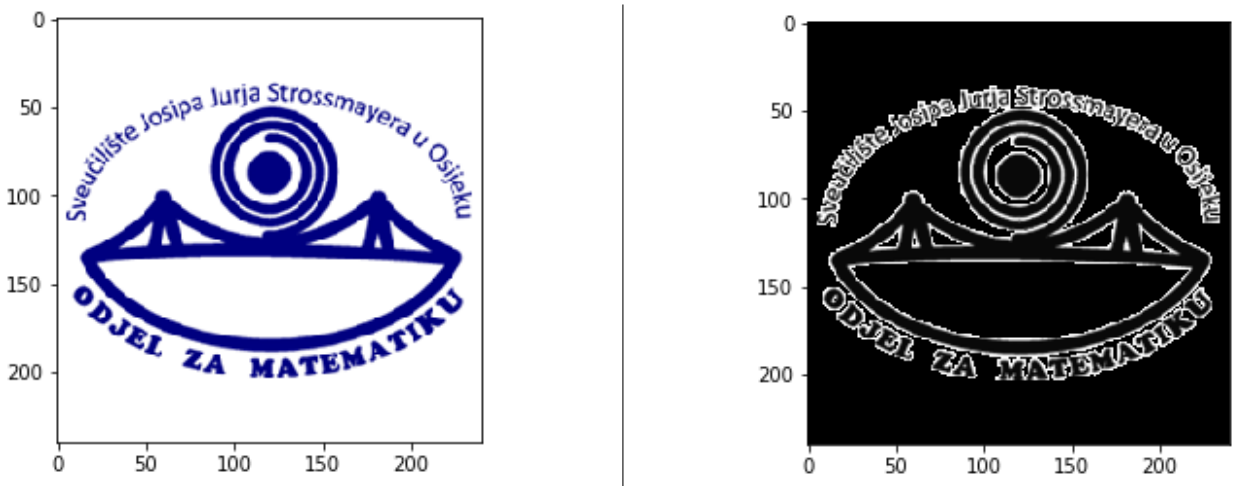
Greyscale je postupak kojim se slika pretvara iz slike u boji u nijanse sive boje. Na primjer za obradu slika u skimage modulu mnoge funkcije upotrebljavaju greyscale slike prije prijenosa, a to se događa jer pojednostavljuje sliku, smanjuje šum (eng. noise reduction) i smanjuje vrijeme obrade jer je na slikama manje informacija.

Postoji nekoliko načina da se izvorna slika pretvori u greyscale sliku. Za primjer ćemo uzeti RGB sliku i piksel $(R, G, B)$

1. Metoda Average
   Average metoda uzima prosječne vrijednosti
   $$gray = \frac{R + G + B}{3}$$
2. Metoda Lightness
   Lightness metoda uzima prosjeke najistaknutijih i najmanje istaknutih boja
   $$gray = \frac{max(R, G, B) + min(R, G, B)}{2}$$
3. Metoda Luminosity
   Luminosity metoda je sofisticiranija inačica Lightness metode također računa prosječne vrijednosti, ali formira takav prosjek za izračun ljudske percepcije. Činjenica je da su ljudi osjetljiviji na zelenu boju u usporedbi na ostale boje pa je iz tog razloga zelena najviše zastupljena. Formula je oblika
   $$gray = 0.21 \times R + 0.71 \times G + 0.07 \times B$$
   Metoda smanjuje kontrast i najbolje funkcionira i to je zadana metoda koja se koristi u većini slučaja.

**Primjer 6.** *Primjena greyscale postupka u Pythonu i spremanje slike*

```
from skimage import io
import matplotlib.pyplot as plt
from skimage import color
image = io.imread("slika.jpg")

#Ručna pretvorba u greyscale
formula = lambda image : np.dot(image[... , :3] , [0.21 , 0.71, 0.07])
gray_image = formula(image)

#Koristimo metodu iz skimage
image =  color.rgb2gray(image)
plt.imsave('nova_slika.png',image )
```




(a) Prije Greyscale-a

(b) Poslje Greyscale-a

Slika 3: Usporedba slika nakon primjene Greyscale-a

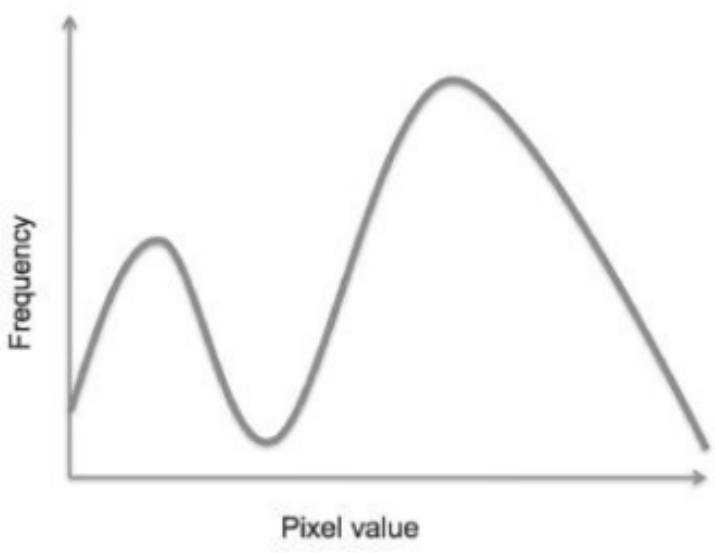
Funkcija imsave() automatski određuje vrstu slike na temelju formata slike koje dajemo. U ovom slučaju, .png ekstenzija uzrokuje spremanje slike kao PNG.

Ovdje je prikazan jedan od mnogo načina za učitavanje, prikazivanje i u konačnici spremanje slika.

## 2.4 Histogram slike

Histogram je grafički prikaz koji prikazuje koliko često se na slici pojavljuju različite vrijednosti boja. Histogram je grafikon s vrijednostima piksela (u rasponu od 0 do 255, ne uvijek) u X-osi i odgovarajućim brojem piksela na slici na Y-osi.

To je samo još jedan način razumijevanja slike. Gledajući histogram slike, dobivamo intuiciju o kontrastu, svjetlini, raspodjeli intenziteta pojedine slike itd. Također je koristan alat u određivanju kvalitete slike i gotovo svi alati za obradu slike nude značajke na histogramu poput Pixlr, ImageJ i CVIPtools.




Slika 4: Slika histograma

Nekoliko zapažanja može se izvesti na osnovu slike:

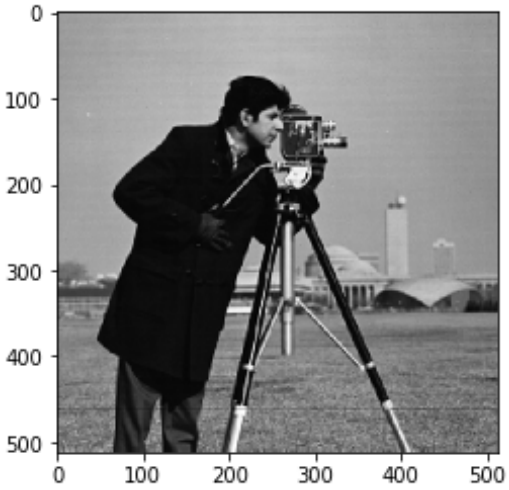
1. Lijeva strana histograma odgovara nižim vrijednostima piksela.

2. Desna strana histograma odgovara višim vrijednostima piksela. Dakle, ako je frekvencija pri višim vrijednostima piksela vrlo visoka, to označava zasićenost.

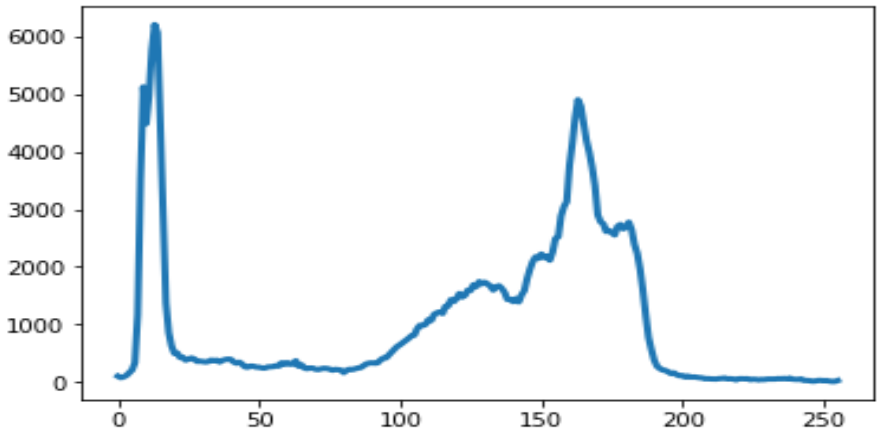
3. Gornji histogram je bi-modalni koji se može koristiti za segmentaciju thresholding. Ali nemaju sve slike bimodalni histogram; stoga postoji mnogo tehnika za segmentaciju pomoću histograma. Nešto više o tome u poglavlju Segmentacije.

**Primjer 7.** *Primjena histograma u Pythonu*

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  from skimage import data
5  from skimage.exposure import histogram
6
7  camera = data.camera()
8  hist, hist_centers = histogram(camera)
9
10 plt.imshow(camera, cmap="gray")
11 plt.show()
12 plt.plot(hist_centers, hist, lw=3)
13 plt.show()
```




Slika 5: Slika kamere




Slika 6: Slika histograma kamere

# 3 Filteri

Do sada je općenito riječ bila o obradi digitalne slike u Pythonu i njegovim znanstvenim modulima koji služe za obradu slike.

U ovom poglavlju temelj razrade je detaljnija obrada slika uporabom određenih filtera za njihovo poboljšanje. Kroz filtriranje slika mogućnost je uklanjanja različitih anomalija kao što je šum (eng. noise), nepoželjnih nečistoća ili isticanja pojedinih dijelova. Python moduli koji se koristite su scikits i scipy.

## 3.1 Filtriranje

Za filtriranje se koristi filter ili maska. Obično je prikazan kao dvodimenzionalan niz odnosno matrica koji se pomiče preko slike i utječe na svaki piksel zasebno. Svaki filter ima svoje karakteristike te je dizajniran tako da ukloni neku vrstu šuma ili pojača određene aspekte na slici. Svaki broj u filteru poznat je kao koficijent. Koficijenti u filteru određuju učinak filtera na sliku. Promotrimo filter u Tablici 1:

| | | |
|---|---|---|
| $F_1$ | $F_2$ | $F_3$ |
| $F_4$ | $F_5$ | $F_6$ |
| $F_7$ | $F_8$ | $F_9$ |

Tablica 1: *Filter* $3x3$

Ako promatramo pojedini piksel kao (i,j) u slici, tada pod-slika oko piksela (i,j) ima istu dimenziju kao dimenzija pripadnog filtera. Centar filtera se poklapa s pikselom (i,j). Pikseli koji se nalaze u pod-slici su pomnoženi s odgovarajućim koficijentima u filteru pa prema tome dobijamo matricu iste veličine kao i filter. Nakon što smo konstruirali takvu matricu kao rezultat vraćamo vrijednost takvu da zbrojimo sve elemente matrice i podjelimo s dimenzijom matrice te tako dobijamo broj koji će zamjeniti vrijednost u pikselu (i,j). Takav postupak klizanja filtera po svakom pikselu slike nazivamo prostorni filter.

Razmotrimo slijedeću pod-sliku sa slike I u središtu (i,j)

| | | |
|---|---|---|
| $I(i-1,\, j-1)$ | $I(i-1,\, j)$ | $I(i-1,\, j+1)$ |
| $I(i,\, j-1)$ | I(i,j) | $I(i,\, j+1)$ |
| $I(i+1,\, j-1)$ | $I(i+1,\, j)$ | $I(i+1,\, j+1)$ |

Tablica 2: *Izgled pod-slike 3x3*

Kombinacija matrice filtera i pod-slike za piksel (i,j)

$$\begin{aligned} I_{novi}(i,j) = {} & F_1 \cdot I(i-1,j-1) + F_2 \cdot I(i-1,j) + F_3 \cdot I(i-1,j+1) \\ & + F_4 \cdot I(i,j-1) + F_5 \cdot I(i,j) + F_6 \cdot I(i,j+1) \\ & + F_7 \cdot I(i+1,j-1) + F_8 \cdot I(i+1,j) + F_9 \cdot I(i+1,j+1) \end{aligned}$$

Kako smo već rekli da se postupak provodi za svaki piksel na slici pa tako uključujući i piksele koji se nalaze na rubovima slike. Kad je filter postavljen na neki od rubnih piksela, dio filtera će biti izvan slike. Budući da vrijednost piksela ne postoji izvan slike moramo ih kreirati. Takav proces stvaranja vrijednosti piksela izvan slike naziva se podstavljanje (eng. padding). Pikseli koji se nalaze izvan slike nazivati ćemo ih padding pikseli njihove vrijednosti možemo postaviti na nulu ili neku konstantnu vrijednost. Padding pikseli se samo kreiraju kod procesa filtriranja te nakon filtriranja se odbacuju. Primjeri nekih od padding piksela prikazani su na Slikama 7,8,9 i 10.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 2 | 5 | 7 | 3 | 10 | 9 |
| 11 | 1 | 4 | 6 | 8 | 2 | 0 |
| 0 | 12 | 10 | 9 | 7 | 4 | 5 |
| 1 | 9 | 7 | 8 | 13 | 11 | 0 |
| 5 | 10 | 14 | 6 | 2 | 1 | 1 |
| 7 | 6 | 11 | 3 | 13 | 8 | 4 |
| 3 | 9 | 6 | 12 | 7 | 10 | 5 |

Slika 7: Slika 7x7

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 2 | 5 | 7 | 3 | 10 | 9 | 0 | 0 |
| 0 | 0 | 11 | 1 | 4 | 6 | 8 | 2 | 0 | 0 | 0 |
| 0 | 0 | 0 | 12 | 10 | 9 | 7 | 4 | 5 | 0 | 0 |
| 0 | 0 | 1 | 9 | 7 | 8 | 13 | 11 | 0 | 0 | 0 |
| 0 | 0 | 5 | 10 | 14 | 6 | 2 | 1 | 1 | 0 | 0 |
| 0 | 0 | 7 | 6 | 11 | 3 | 13 | 8 | 4 | 0 | 0 |
| 0 | 0 | 3 | 9 | 6 | 12 | 7 | 10 | 5 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Slika 8: Padding s nulom

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 5 | 5 | 0 | 2 | 5 | 7 | 3 | 10 | 9 | 5 | 5 |
| 5 | 5 | 11 | 1 | 4 | 6 | 8 | 2 | 0 | 5 | 5 |
| 5 | 5 | 0 | 12 | 10 | 9 | 7 | 4 | 5 | 5 | 5 |
| 5 | 5 | 1 | 9 | 7 | 8 | 13 | 11 | 0 | 5 | 5 |
| 5 | 5 | 5 | 10 | 14 | 6 | 2 | 1 | 1 | 5 | 5 |
| 5 | 5 | 7 | 6 | 11 | 3 | 13 | 8 | 4 | 5 | 5 |
| 5 | 5 | 3 | 9 | 6 | 12 | 7 | 10 | 5 | 5 | 5 |
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |

Slika 9: Padding s konstantom 5

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 2 | 5 | 7 | 3 | 10 | 9 | 9 | 10 |
| 0 | 0 | 0 | 2 | 5 | 7 | 3 | 10 | 9 | 9 | 9 |
| 11 | 11 | 11 | 1 | 4 | 6 | 8 | 2 | 0 | 0 | 0 |
| 0 | 0 | 0 | 12 | 10 | 9 | 7 | 4 | 5 | 5 | 5 |
| 1 | 1 | 1 | 9 | 7 | 8 | 13 | 11 | 0 | 0 | 0 |
| 5 | 5 | 5 | 10 | 14 | 6 | 2 | 1 | 1 | 1 | 1 |
| 7 | 7 | 7 | 6 | 11 | 3 | 13 | 8 | 4 | 4 | 4 |
| 3 | 3 | 3 | 9 | 6 | 12 | 7 | 10 | 5 | 5 | 5 |
| 3 | 3 | 3 | 9 | 6 | 12 | 7 | 10 | 5 | 5 | 5 |

Slika 10: Padding s najbližim susjedom

## 3.2 Median filter

Median filter je jedan od najpopularnijih nelinearnih filtera. Vrijednost pojedinog piksela (i,j) na slici uz pomoć Median filtera se računa tako da se napravi median vrijednosti koficijenata u pod-slici. Kao za primjer uzima se niz brojeva.

$$5, 2, 9, 13, 19, 17, 4$$

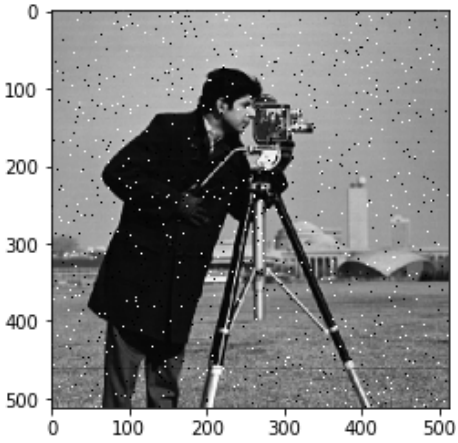
Za izračunavanje vrijednosti medijana vrijednosti su raspoređene uzlaznim redosljedom, dakle novi niz brojeva izgleda.

$$2, 4, 5, 9, 13, 17, 19$$

Medijan je vrijednost koja niz djeli na dvije jednake polovice; u ovom slučaju to je broj 9. Dakle pojediniom pikselu (i,j) dodjeljena vrijednost će biti 9 na filtliranoj slici. Median filter se koristi za uklanjanje šuma (eng. noise) koji se karakterizira kao crne i bijele mrlje (eng. salt-and-papper noise) nasumično raspoređene na slici.

**Primjer 8.** *Primjena Median filtera u Pythonu:*

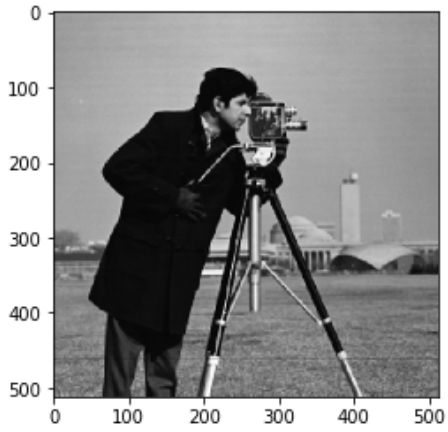
```
#Potrebne biblioteke
import numpy as np
import matplotlib.pyplot as plt
from skimage import data
from skimage.morphology import disk
from skimage import filters

#Ucitavamo sliku u program
image = data.camera()

#Dodajemo buku u sliku
noise = np.random.random(image.shape)
noisy_image = image
noisy_image[noise > 0.99] = 255
noisy_image[noise < 0.01] = 0
#Prikazujemo sliku
plt.imshow(noisy_image, cmap="gray")
```




Slika 11: Buka na slici (eng.noise)

**Primjer 9.** *Primjena Median filtera na Sliku 11:*

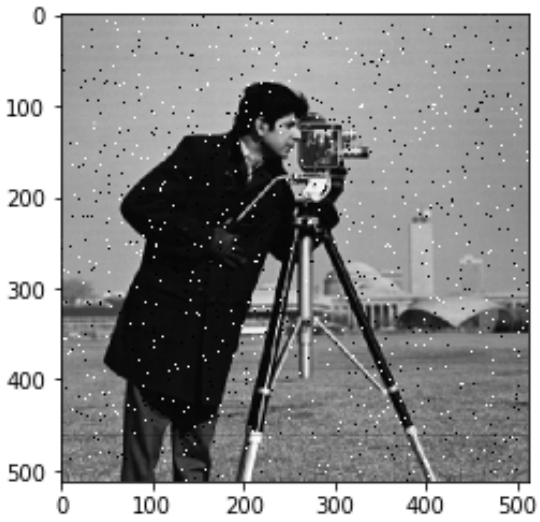
```
#Potrebne biblioteke
import numpy as np
import matplotlib.pyplot as plt
from skimage import data
from skimage.morphology import disk
from skimage import filters

size = 1
#Primjena Median filtera
image_filter = filters.median(noisy_image, disk(size))
#Prikaz filtrirane slike
plt.imshow(image_filter, cmap="gray")
```



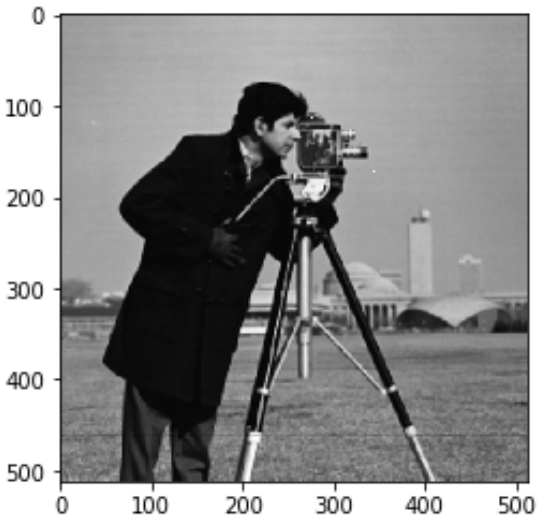

Slika 12: Filtrirana slika

Median filter je učinkovito uklonio šum prikazanu na Slici 12. U gore navedenom kodu korišten je $size = 1$ odnosno veličinu koja predstavlja filter (masku) veličine $1 \times 1$.

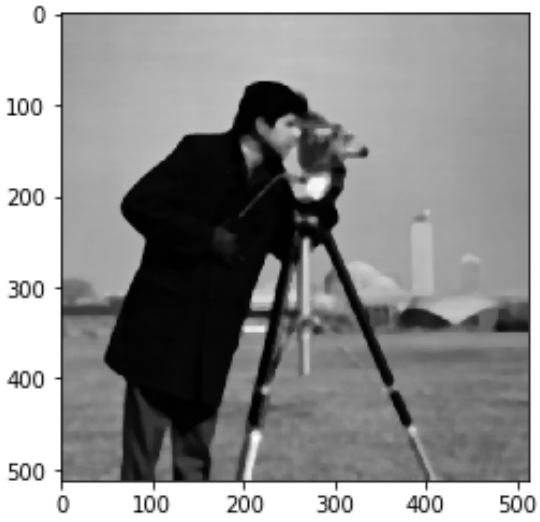
Utjecaj promjena dimenzije filtera prikazani su na Slikama 14,15 i 16.



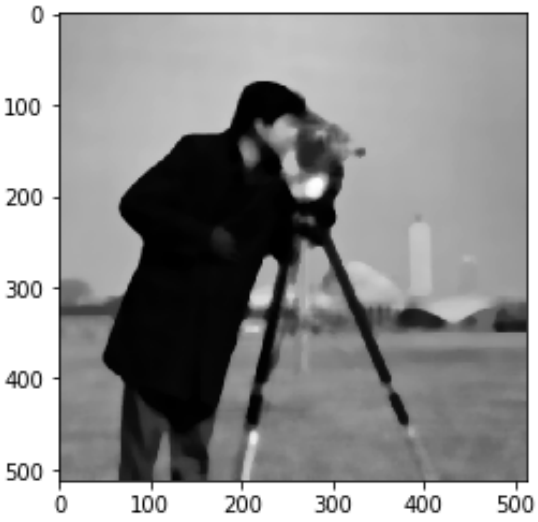

Slika 13: Slika s bukom




Slika 14: $size = 1$




Slika 15: $size = 5$




Slika 16: $size = 8$

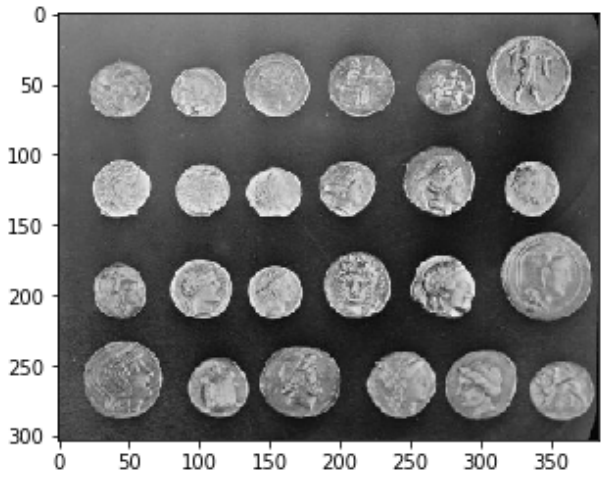
## 3.3 Max filter

Max filter pojačava svijetle točke na određenoj slici. U Median filteru računat je median, a u Max filteru računa se maksimalna vrijednost u pod-slici i zamjenjuje vrijednost piksela (i,j). Python funkcija za Max filter ima iste argumente kao i za medijan filter kojega smo naveli gore.

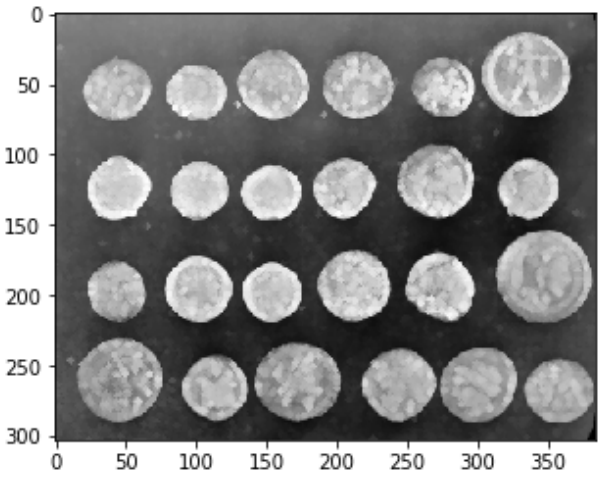
**Primjer 10.** *Primjena Max filtera u Pythonu:*

```
#Potrebne biblioteke
import numpy as np
import matplotlib.pyplot as plt
from skimage import data
from skimage.morphology import disk
from skimage.filters.rank import maximum

#Ucitavamo sliku
image = data.coins()

size=2
#Primjena max filtera
image_filter= maximum(image, disk(size))
#Prikazujemo sliku
plt.imshow(image_filter, cmap="gray")
```

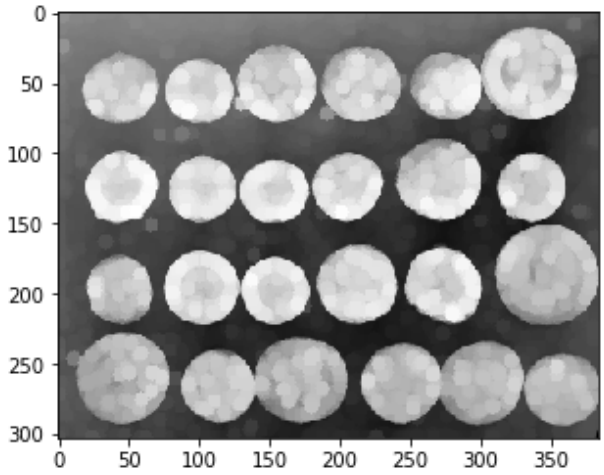
Učinak promjena dimenzije filtera prikazam je na Slici 18. 19 i 20.



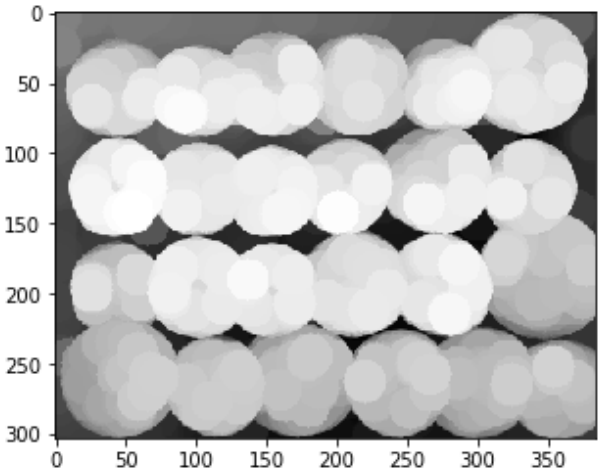

Slika 17: Orginalna slika




Slika 18: $size = 2$




Slika 19: $size = 5$

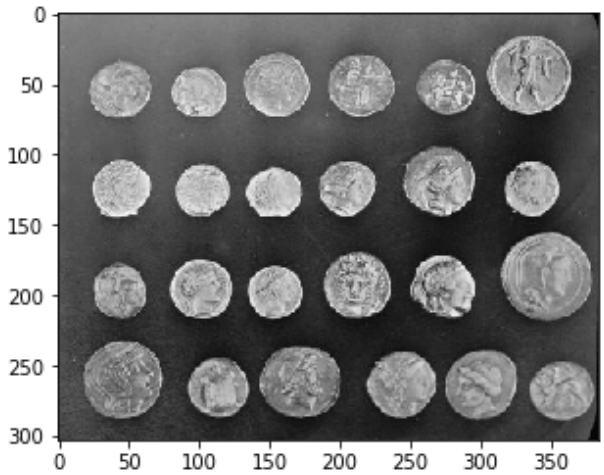



Slika 20: $size = 15$

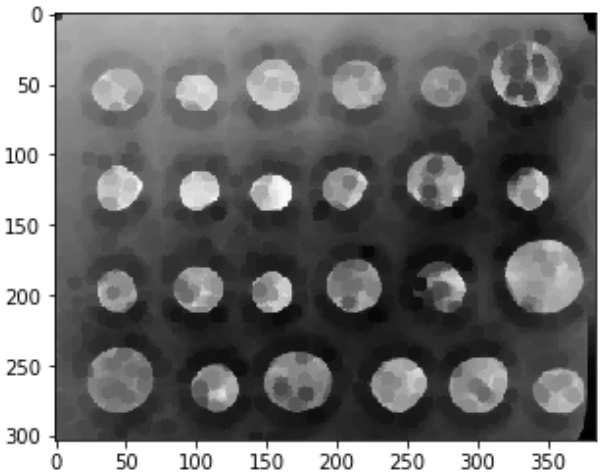
## 3.4 Min filter

Min filter radi suprotno od Max filtera odnosno on pojačava tamnije točke na slici. Vrijednost u pod-slici zamjenjujemo s minimalnom vrijednosti i također ima iste argumente kao i za medijan filter.

Primjena Min filtera prikazana je na Slici 21 i 22.




Slika 21: Orginalna slika




Slika 22: Primjena Min filtera

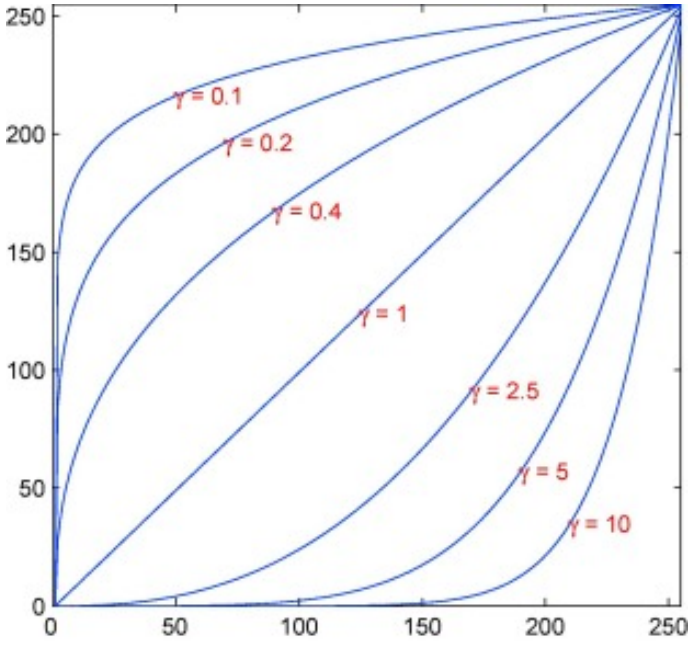
## 3.5 Gama korekcija

U prethodnim poglavljima razrađeno je djelovanje određenih filtera na sliku i pojačavanje kvalitete slike tako da se važni detalji mogu vizualizirati i kvantificirati. Postoje druge tehnike za poboljšavanje slika u smislu transformiranja ulazne vrijednosti piksela u nove vrijednosti u izlaznoj slici pomoću funkcija za preslikavanje.

Jedna od takvih tehnika je gama korekcija koja služi za poboljšanje kvalitete slike. Gama korekcija za piksel (i,j) je dana izrazom,

$$t(i,j) = k \cdot I(i,j)^{\gamma} \tag{1}$$

gdje su k i $\gamma$ pozitivne konstante, a I(i,j) vrijednost intenziteta piksela na slici. Većinom k = 1, a kada je $\gamma = 1$ preslikavanje je linearno a izlazna slika je jednaka slici koju je proslijeđena. U slučaju kada je $\gamma < 1$ tada uzak raspon tamnijih ili niskih vrijednosti piksela u orginalnoj slici preslikavaju se u široki raspon intenziteta na izlaznoj slici, dok široki raspon svijetlih ili visokih intenziteta piksela na ulaznoj slici preslikavaju se u uski raspon visokog intenziteta na izlaznoj slici. Za $\gamma > 1$ suprotno je od vrijednosti $\gamma < 1$. Ovisno o vrijednosti game, na slici se istovremeno pojačava ili samo viskoi ili samo niski intenzitet piksela.




Slika 23: Graf gamma korekcije za različiti $\gamma$

Ljudski mozak također koristi gama korekciju za obradu slike, dakle gamma korekcija je ugrađena značajka u svim uređajima koji prikazuju sliku poput računalnih monitora i televiziskih ekrana koji sadržavaju gamma korekciju tako da prikazuju najbolji kontrast slike na zaslonu.

Kako u 8-bitnoj slici vrijednosti intenziteta piksela kreću se od 0 do 255. Ako se transformacija primjenjuje prema jednadžbi (1), a za $\gamma > 1$ intenzitet izlaznih piksela će biti izvan granice. Da bi se to spriječilo potrebno je normalizirati intenzitet piksela I(i,j).

$$I_{norm} = \frac{I(i,j)}{max(I)} \tag{2}$$

Za $k = 1$ i zamjenjujući I(i,j) s $I_{norm}$ zatim primjenom prirodnog logaritma s obje strane dobijamo jednadžbu

$$ln(t(i,j)) = ln(I_{norm})^{\gamma} = \gamma \cdot ln(I_{norm}) \tag{3}$$

ekivalentna jednadžba

$$e^{ln(t(i,j))} = e^{\gamma \cdot ln(I_{norm})} \tag{4}$$

kako je $e^{ln(x)} = x$ lijevu stranu jednadžbe (4) možemo zapisati

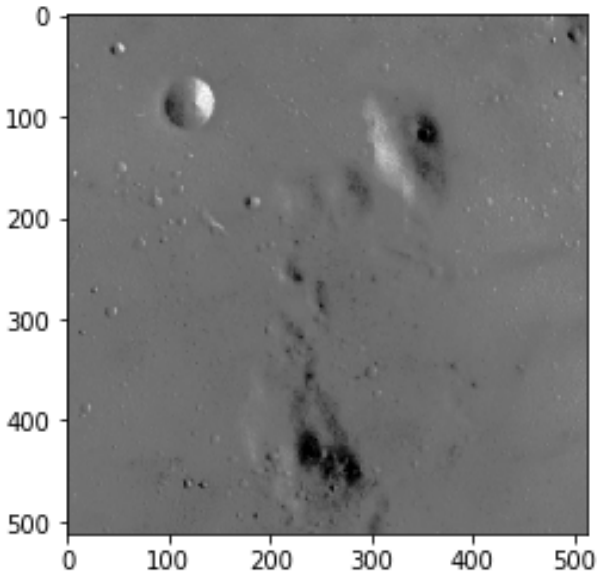
$$t(i,j) = e^{\gamma \cdot ln(I_{norm})} \tag{5}$$

da imamo izlaz u rasponu od 0 do 255 pomnožimo desnu stranu s 255 i dobijamo jednadžbu.

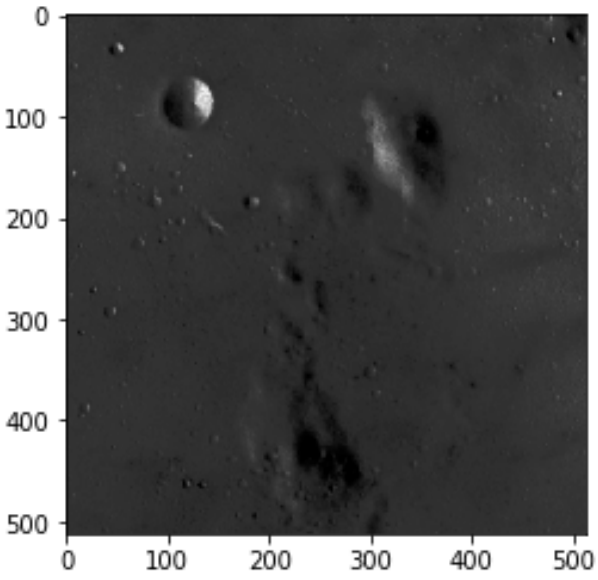
$$t(i,j) = e^{\gamma \cdot ln(I_{norm})} \cdot 255 \tag{6}$$

**Primjer 11.** *Primjena gama korekcije u Pythonu:*

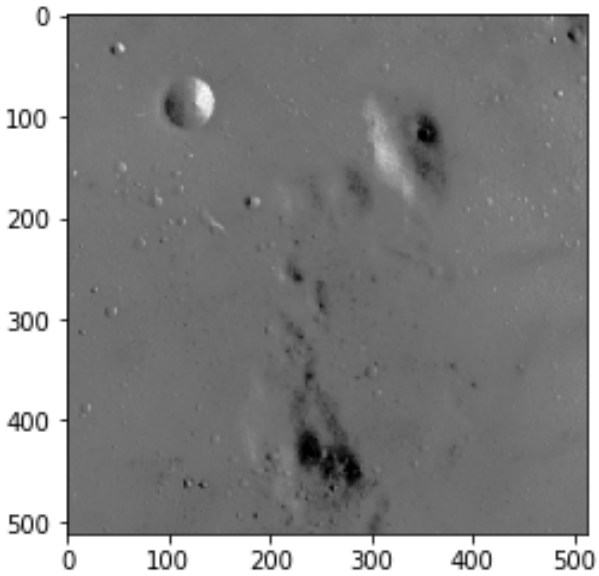
```python
#Potrebne biblioteke
import numpy as np
import matplotlib.pyplot as plt
from skimage import data
from skimage import exposure

#Ucitavamo sliku
image = data.moon()

#Gamma korekcija
gamma = 2
gamma_corrected = exposure.adjust_gamma(image, gamma)

#Prikaz slike
plt.imshow(gamma_corrected, cmap="gray")
```



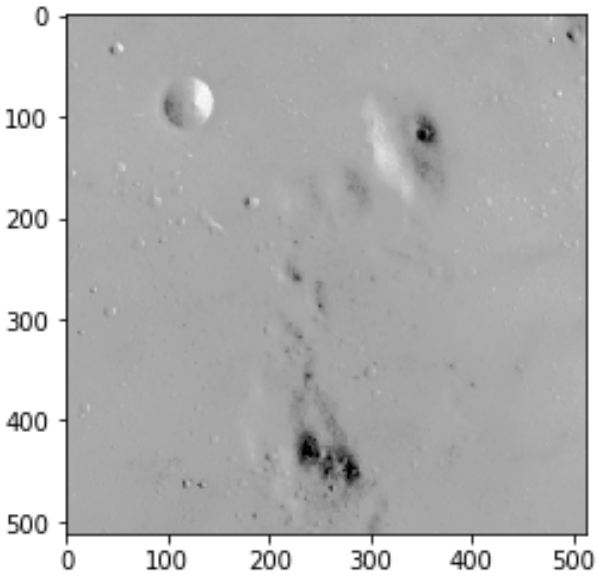

Slika 24: Orginalna slika



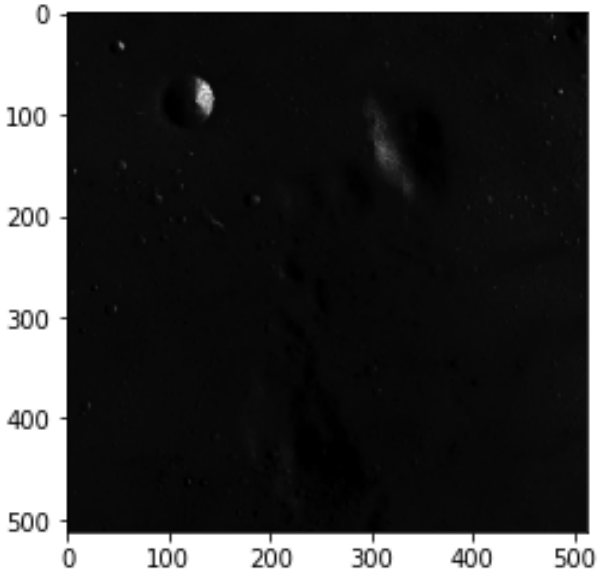

Slika 25: $\gamma = 2$



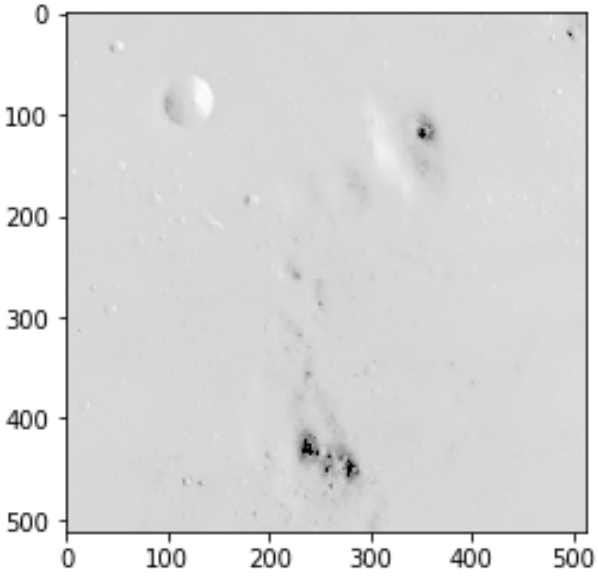

Slika 26: Orginalna slika




Slika 27: $\gamma = 0.50$




Slika 28: $\gamma = 4$




Slika 29: $\gamma = 0.20$

# 4 Segmentacija

Segmentacija je proces razdvajanja slike na višestruke logičke segmente. Segmenti se definiraju kao skupine piksela koje dijele zajedničke karakteristike poput intenziteta piksela, teksture itd. Postoje brojne metode segmentacije koje mogu se klasificirati kao:

- Segmentacija na temelju histograma
- Segmentacija na temelju regije
- Segmentacija ruba
- Metoda diferencijalne jednadžbe

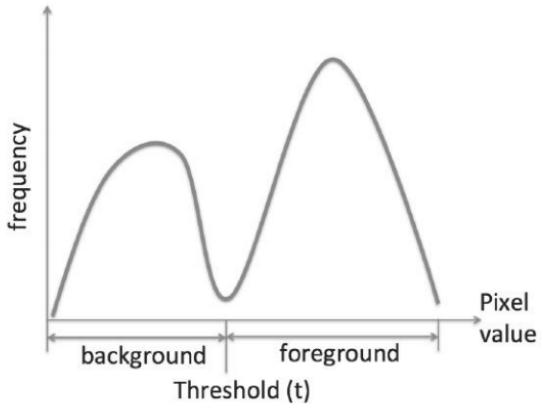
U ovom radu obraditi će se segmentacija na temelju histograma , ostale metode nisu predmetom ovoga rada.

## 4.1 Segmentacija na temelju histograma

Prvo, potrebno je definirati Thresholding. Thresholding je najjednostavniji način segmentacije predmeta iz pozadine. U metodi koja se temelji na histogramu (Slika 30.) utvrđuje se vrijednost praga (eng. threshold). Svaki piksel sa slike se uspoređuje s vrijednosti praga. Ako je intenzitet piksela manji od vrijednosti praga pripadajućem pikselu bit će dodijeljena vrijednost nula u segmentiranoj slici. U suprotnom ako je intenzitet piksela veći od vrijednosti praga dodijeljena vrijednost će biti jedan u segmentiranoj slici.

**if** ps $\geq$ threshold **then**
  pseg = 1
**else**
  pseg = 0
**end if**

gdje je ps vrijednost piksela na slici, a pseg vrijednost piksela u segmentiranoj slici.




Slika 30: Podjela praga(eng. threshold) piksela

### 4.1.1 Otsu's metoda

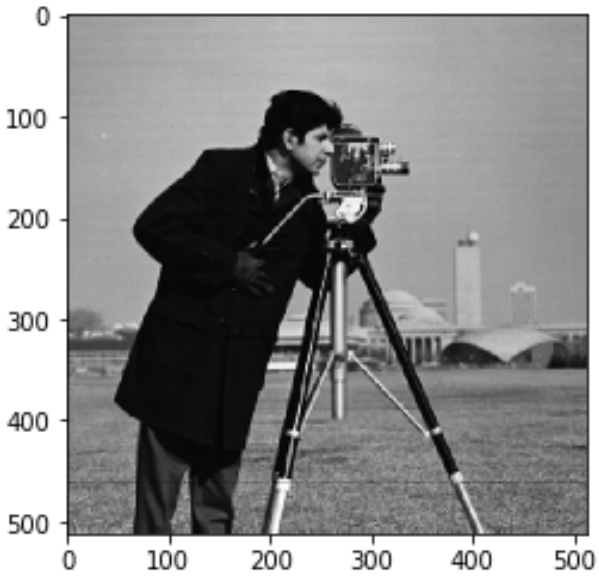
Postoje različite metode segmentacije koje se razlikuju u svojim tehnikama određivanja praga (eng. threshold). A jedna od takvih metoda je i Otsu's metoda.

Osnovna ideja je razdvajanje histograma slike na dva klastera s pragom. Otsu's metoda traži vrijednost praga tako da maksimizira varijancu između dva klastera odnosno background i foreground.

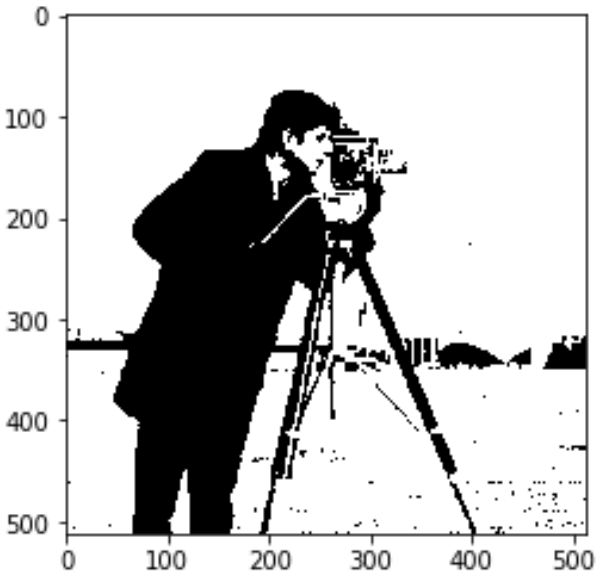
Ideja algoritma:

1. Učitati sliku
2. Dobivanje histograma slike (distribucija piksela)
3. Izračunavanje vrijednosti praga t
4. Zamijena piksela slike bijelim bojama u onim područjima gdje je intenzitet piksela veća od t, a u crnu u suprotnom slučaju

```
#Potrebne biblioteke
import numpy as np
import matplotlib.pyplot as plt
from skimage import data
from skimage import filters

#Ucitavamo sliku
image = data.camera()

#Vrijednost praga
threshold= filters.threshold_otsu(image)

#Zamjena piksela
thresholded = image > threshold

#Prikaz slike
plt.imshow(thresholded, cmap="gray")
```

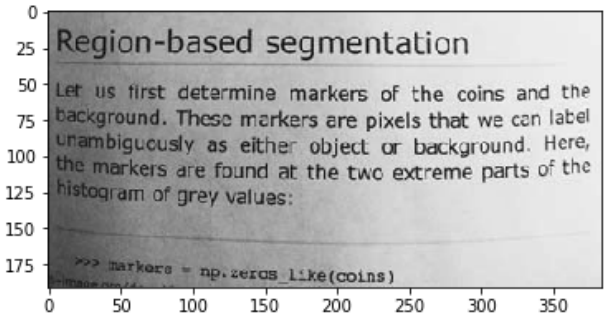


Slika 31: Orginalna slika

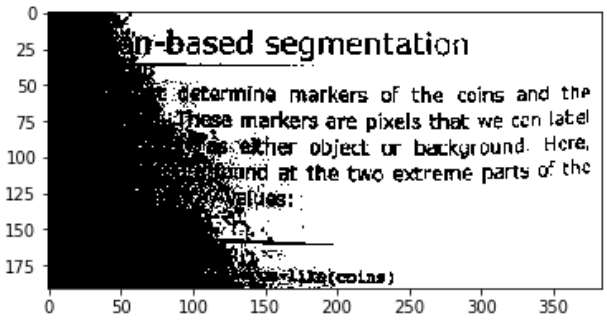


Slika 32: Otsu's metoda

Otsu's metoda metoda koristi histogram za određivanje praga i stoga uvelike ovisi o ulaznoj slici. Primjer kada Otsu's metoda zbog sjene na ulaznoj slici (Slika 33.) nije segmentirala točno (Slika 34.). Kako ne bi ovisili uvijek o ulaznoj slici koristimo metodu Adaptivnih thresholdinga koji pomažu u rješavanju takvih problema.




Slika 33: Orginalna slika



Slika 34: Otsu's metoda koja nije uspjela točno segmentirati

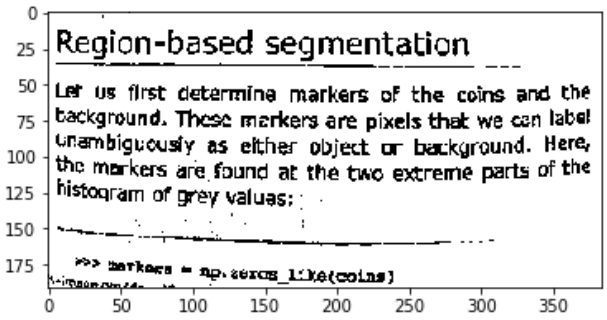
## 4.2 Adaptivni Thresholding

U prethodnom odjeljku za prag vrijednost korištena je globalna vrijednost. Ponekad je to krivi pristup računanja iz razloga što slika ima različite uvjete osvjetljenja u različitim područjima. U tom slučaju koristimo adaptivni thresholding koji izračunava prag za male regije slike i na taj način dobivamo različite pragove za različite regije iste slike i to daje bolje rezultate za slike s različitim osvjetljenjem.

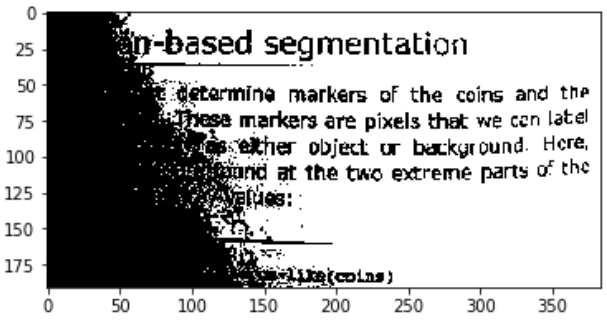
U adaptivnom thresholdingu slika je prvo podijeljena u male pod-slike. Vrijednost praga za svaku pod-sliku izračunava se i koristi za segmentiranje slike. Vrijednost praga za pod-slike se može izračunati različitim metodama poput srednja (eng. mean), medijan (eng. median) ili Gaussian metoda.

U slučaju metode srednje vrijednosti za prag se koristi srednja vrijednost slike, dok je za medijan metodu medijan. Također možemo koristiti prilagođene formule koje se mogu koristiti za izračun praga, na primjer, u pod-slicise može upotrijebiti prosjek maksimalnih i minimalnih vrijednosti piksela.

**Primjer 12.** *Usporedba Adaptivnog thresholdinga i Otsu's metode*




Slika 35: Adaptivni Thresholding



Slika 36: Otsu's metoda

# Literatura

[1] R. Chityala, S. Pudipeddi, *Image Processing and Acquisition using Python*, University of Minnesota at Minneapolis USA.

[2] P.K. Sinha , *Image Acquisition and Preprocessing for Machine Vision Systems* Bellingham, Washington USA.

[3] Khan M. Iftekharuddin, Abdul A. Awwal, *Image Processing*

[4] *Sckit-image*, https://scikit-image.org/

[5] *Filters*, https://scikit-image.org/docs/dev/api/skimage.filters.html

[6] *Segmentacija*, https://scikit-image.org/docs/dev/api/skimage.segmentation.html